EXHIBIT A

US009407564B2

(12) **United States Patent**
Major et al.

(10) **Patent No.: US 9,407,564 B2**
(45) **Date of Patent: *Aug. 2, 2016**

(54) **APPARATUS, SYSTEM, AND METHOD FOR ADAPTIVE-RATE SHIFTING OF STREAMING CONTENT**

(71) Applicant: **ECHOSTAR TECHNOLOGIES L.L.C.**, Englewood, CO (US)

(72) Inventors: **Robert Drew Major**, Orem, UT (US); **Mark B. Hurst**, Cedar Hills, UT (US)

(73) Assignee: **EchoStar Technologies L.L.C.**, Englewood, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **14/516,303**

(22) Filed: **Oct. 16, 2014**

(65) **Prior Publication Data**

US 2015/0039782 A1 Feb. 5, 2015

Related U.S. Application Data

(63) Continuation of application No. 11/116,783, filed on Apr. 28, 2005, now Pat. No. 8,868,772.

(60) Provisional application No. 60/566,831, filed on Apr. 30, 2004.

(51) **Int. Cl.**
H04L 12/853 (2013.01)
H04L 12/825 (2013.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04L 47/25*** (2013.01); ***H04L 65/60*** (2013.01); ***H04N 21/25808*** (2013.01);
(Continued)

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,535,355	A	8/1985	Arn et al.
5,168,356	A	12/1992	Acampora et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CA	2466482	5/2003
EP	0 711 077 A2	5/1996

(Continued)

OTHER PUBLICATIONS

Bill Birney, Intellegent Streaming, May 2003, all pages.*

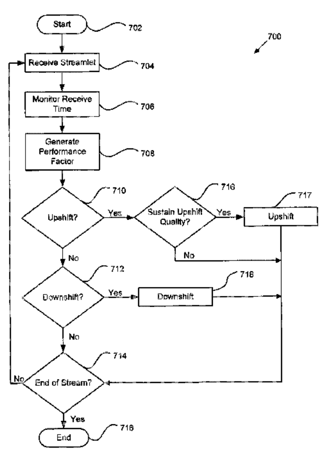
(Continued)

Primary Examiner — Ninos Donabed
(74) *Attorney, Agent, or Firm* — Ingrassia Fisher & Lorenz, P.C.

(57) **ABSTRACT**

An apparatus for adaptive-rate shifting of streaming content includes an agent controller module configured to simultaneously request at least portions of a plurality of streamlets. The agent controller module is further configured to continuously monitor streamlet requests and subsequent responses, and accordingly request higher or lower quality streamlets. A staging module is configured to stage the streamlets and arrange the streamlets for playback on a content player. A system includes a data communications network, a content server coupled to the data communications network and having a content module configured to process content and generate a plurality of high and low quality streams, and the apparatus. A method includes simultaneously requesting at least portions of a plurality of streamlets, continuously monitoring streamlet requests and subsequent responses, and accordingly requesting higher or lower quality streamlets, and staging the streamlets and arranging the streamlets for playback on a content player.

16 Claims, 7 Drawing Sheets



(51) **Int. Cl.**

H04N 21/258	(2011.01)
H04N 21/2662	(2011.01)
H04N 21/643	(2011.01)
H04N 21/647	(2011.01)
H04N 21/84	(2011.01)
H04N 21/845	(2011.01)
H04L 29/06	(2006.01)

(52) **U.S. Cl.**

CPC ***H04N 21/2662*** (2013.01); ***H04N 21/643*** (2013.01); ***H04N 21/64769*** (2013.01); ***H04N 21/64792*** (2013.01); ***H04N 21/84*** (2013.01); ***H04N 21/845*** (2013.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,267,334 A	11/1993	Normille et al.	
5,404,446 A	4/1995	Bowater et al.	
5,768,527 A	6/1998	Zhu et al.	
5,841,432 A	11/1998	Carmel et al.	
5,953,506 A	9/1999	Kalra et al.	
6,091,775 A	7/2000	Hibi et al.	
6,091,777 A	7/2000	Guetz et al.	
6,122,660 A	9/2000	Baransky et al.	
6,185,736 B1	2/2001	Ueno	
6,195,680 B1	2/2001	Goldszmidt et al.	709/203
6,366,614 B1	4/2002	Pian et al.	375/240.02
6,374,289 B2	4/2002	Delaney et al.	
6,389,473 B1	5/2002	Carmel et al.	
6,449,719 B1	9/2002	Baker	
6,486,803 B1	11/2002	Luby et al.	
6,490,627 B1	12/2002	Kalra et al.	
6,510,553 B1	1/2003	Hazra	
6,552,227 B2	4/2003	Mendelovici et al.	
6,574,591 B1	6/2003	Kleiman et al.	
6,604,118 B2	8/2003	Kleiman et al.	
6,618,752 B1	9/2003	Moore et al.	
6,654,790 B2	11/2003	Ogle et al.	
6,675,199 B1	1/2004	Mohammed et al.	
6,697,072 B2	2/2004	Russell et al.	
6,721,723 B1	4/2004	Gibson et al.	
6,731,600 B1	5/2004	Patel et al.	
6,732,183 B1	5/2004	Graham	
6,760,772 B2	7/2004	Zou et al.	
6,792,449 B2	9/2004	Colville et al.	
6,795,863 B1	9/2004	Doty, Jr.	
6,801,947 B1	10/2004	Li	
6,845,107 B1	1/2005	Kitazawa et al.	
6,850,965 B2	2/2005	Allen	
6,859,839 B1	2/2005	Zahorjan et al.	
6,874,015 B2	3/2005	Kaminsky et al.	
6,885,471 B1	4/2005	Minowa et al.	
6,968,387 B2	11/2005	Lanphear	
6,976,090 B2	12/2005	Ben-Shaul et al.	
7,031,700 B1	4/2006	Weaver et al.	
7,046,805 B2	5/2006	Fitzhardinge et al.	
7,054,365 B2	5/2006	Kim et al.	
7,054,774 B2	5/2006	Batterberry et al.	
7,054,911 B1	5/2006	Lango et al.	
7,075,986 B2	7/2006	Girod et al.	
7,093,001 B2	8/2006	Yang et al.	
7,096,271 B1	8/2006	Omoigui et al.	
7,099,954 B2	8/2006	Li et al.	
7,111,044 B2	9/2006	Lee	
7,116,894 B1	10/2006	Chatterton	
7,124,164 B1	10/2006	Chemtob	
7,174,385 B2	2/2007	Li	
7,176,957 B2	2/2007	Ivashin et al.	
7,177,642 B2	2/2007	Sanchez Herrero et al.	
7,190,670 B2	3/2007	Varsa et al.	
7,194,549 B1	3/2007	Lee et al.	
7,240,100 B1	7/2007	Wein et al.	
7,260,640 B1	8/2007	Kramer et al.	
7,274,740 B2	9/2007	van Beek et al.	
7,295,520 B2	11/2007	Lee et al.	
7,310,678 B2	12/2007	Gunaseelan et al.	
7,313,236 B2	12/2007	Amini et al.	
7,325,073 B2	1/2008	Shao et al.	
7,328,243 B2	2/2008	Yeager et al.	
7,330,908 B2	2/2008	Jungck	
7,334,044 B1	2/2008	Allen	
7,349,358 B2	3/2008	Hennessey et al.	
7,349,976 B1	3/2008	Glaser et al.	
7,369,610 B2	5/2008	Xu et al.	
7,376,747 B2	5/2008	Hartop	
7,391,717 B2	6/2008	Klemets et al.	
7,408,984 B2	8/2008	Lu et al.	
7,412,531 B1	8/2008	Lango et al.	
7,477,688 B1	1/2009	Zhang et al.	
7,523,181 B2	4/2009	Swildens et al.	
7,529,541 B2	5/2009	Cho et al.	
7,536,469 B2	5/2009	Chou et al.	
7,546,355 B2	6/2009	Kalnitsky	
7,555,464 B2	6/2009	Candelore	
7,558,472 B2	7/2009	Locket et al.	
7,558,869 B2 *	7/2009	Leon	H04L 29/06027 370/235
7,577,750 B2	8/2009	Shen et al.	
7,593,333 B2	9/2009	Li et al.	
7,599,307 B2	10/2009	Seckin et al.	
7,609,652 B2	10/2009	Kellerer et al.	
7,631,039 B2	12/2009	Eisenberg	
7,653,735 B2	1/2010	Mandato et al.	
7,657,644 B1	2/2010	Zheng	
7,660,906 B1	2/2010	Armour	
7,707,303 B2 *	4/2010	Albers	H04L 1/1835 709/231
7,719,985 B2	5/2010	Lee et al.	
7,733,830 B2	6/2010	Curcio et al.	
7,760,801 B2	7/2010	Ghanbari et al.	
7,761,609 B1	7/2010	Srinivasan et al.	
7,779,135 B2	8/2010	Hudson et al.	
7,788,395 B2	8/2010	Bowra et al.	
7,797,439 B2	9/2010	Cherkasova et al.	
7,817,985 B2	10/2010	Moon	
7,818,444 B2	10/2010	Brueck et al.	
7,873,040 B2	1/2011	Karlsgodt	
8,036,265 B1	10/2011	Reynolds et al.	
8,135,852 B2	3/2012	Nilsson et al.	
8,209,429 B2	6/2012	Jacobs et al.	
8,370,514 B2	2/2013	Hurst et al.	
8,402,156 B2	3/2013	Brueck et al.	
8,612,624 B2	12/2013	Frueck et al.	
8,683,066 B2	3/2014	Hurst et al.	
8,880,721 B2	11/2014	Hurst et al.	
2001/0013128 A1	8/2001	Hagai et al.	
2001/0047423 A1	11/2001	Shao et al.	
2002/0073167 A1	6/2002	Powell et al.	
2002/0087634 A1	7/2002	Ogle et al.	
2002/0091840 A1	7/2002	Pulier et al.	
2002/0097750 A1	7/2002	Gunaseelan et al.	
2002/0118809 A1	8/2002	Eisenberg	
2002/0122491 A1	9/2002	Karczewicz et al.	
2002/0131496 A1	9/2002	Vasudevan et al.	
2002/0133547 A1	9/2002	Lin	
2002/0136406 A1	9/2002	Fitzhardinge et al.	
2002/0138619 A1	9/2002	Ramaley et al.	
2002/0144276 A1	10/2002	Radford et al.	
2002/0146102 A1	10/2002	Lang	
2002/0152317 A1	10/2002	Wang et al.	
2002/0152318 A1	10/2002	Menon et al.	
2002/0161898 A1	10/2002	Hartop et al.	
2002/0161908 A1 *	10/2002	Benitez	G06F 8/65 709/231
2002/0161911 A1	10/2002	Pinckney, III et al.	
2002/0169926 A1	11/2002	Pinckney, III et al.	
2002/0174434 A1	11/2002	Lee et al.	
2002/0176418 A1	11/2002	Hunt et al.	
2002/0178138 A1	11/2002	Ender et al.	
2002/0178330 A1	11/2002	Schlowsky-Fischer et al.	
2002/0184391 A1	12/2002	Phillips	
2002/0188745 A1 *	12/2002	Hughes	H04L 29/06027 709/231

(56) **References Cited**

U.S. PATENT DOCUMENTS

2002/0194608 A1	12/2002	Goldhor	
2003/0005455 A1	1/2003	Bowers	
2003/0007464 A1	1/2003	Balani	
2003/0014684 A1	1/2003	Kashyap	
2003/0018966 A1	1/2003	Cook et al.	
2003/0021166 A1	1/2003	Soloff	
2003/0021282 A1*	1/2003	Hospodor	H04L 12/5695 370/401
2003/0037103 A1	2/2003	Salmi et al.	
2003/0065803 A1	4/2003	Heuvelman	
2003/0067872 A1	4/2003	Harrell et al.	
2003/0078972 A1	4/2003	Tapissier et al.	
2003/0081582 A1	5/2003	Jain et al.	
2003/0093790 A1	5/2003	Logan et al.	
2003/0107994 A1	6/2003	Jacobs et al.	
2003/0135631 A1	7/2003	Li et al.	
2003/0140159 A1	7/2003	Campbell et al.	
2003/0151753 A1	8/2003	Li et al.	358/1.9
2003/0152036 A1	8/2003	Quigg Brown et al.	
2003/0154239 A1	8/2003	Davis et al.	
2003/0195977 A1*	10/2003	Liu	H04L 29/06027 709/231
2003/0204519 A1	10/2003	Sirivara et al.	
2003/0204602 A1	10/2003	Hudson et al.	
2003/0220972 A1	11/2003	Montet et al.	
2004/0003101 A1	1/2004	Roth et al.	
2004/0010613 A1	1/2004	Apostolopoulos et al.	
2004/0030547 A1	2/2004	Leaning et al.	
2004/0030599 A1	2/2004	Sie et al.	
2004/0030797 A1	2/2004	Akinlar et al.	
2004/0031054 A1	2/2004	Dankworth et al.	
2004/0049780 A1	3/2004	Gee	
2004/0054551 A1	3/2004	Ausubel et al.	
2004/0071209 A1	4/2004	Burg et al.	
2004/0083283 A1	4/2004	Sundaram et al.	
2004/0093420 A1	5/2004	Gamble	
2004/0098748 A1	5/2004	Bo et al.	
2004/0103444 A1	5/2004	Weinberg et al.	
2004/0117427 A1	6/2004	Allen et al.	
2004/0143672 A1	7/2004	Padmanabham et al.	
2004/0153458 A1	8/2004	Noble et al.	
2004/0168052 A1	8/2004	Clisham et al.	
2004/0170392 A1	9/2004	Lu et al.	
2004/0179032 A1*	9/2004	Huang	H04N 19/61 715/723
2004/0199655 A1*	10/2004	Davies	H04L 47/10 709/231
2004/0220926 A1	11/2004	Lamkin et al.	
2004/0221088 A1*	11/2004	Lisitsa	H04L 29/06027 710/316
2004/0260701 A1	12/2004	Lehikoinen et al.	
2004/0267956 A1*	12/2004	Leon	H04L 12/5602 709/231
2005/0009520 A1	1/2005	Herrero et al.	
2005/0015509 A1	1/2005	Sitaraman	
2005/0024487 A1	2/2005	Chen	
2005/0033855 A1	2/2005	Moradi et al.	
2005/0050152 A1	3/2005	Penner et al.	
2005/0055425 A1	3/2005	Lango et al.	
2005/0066063 A1	3/2005	Grigorovitch et al.	
2005/0076136 A1	4/2005	Cho et al.	
2005/0084166 A1	4/2005	Bonch et al.	
2005/0108414 A1	5/2005	Taylor et al.	709/231
2005/0120107 A1	6/2005	Kagan et al.	
2005/0123058 A1	6/2005	Greenbaum et al.	
2005/0185578 A1	8/2005	Padmanabhan et al.	
2005/0188051 A1	8/2005	Sneh	
2005/0204046 A1	9/2005	Watanabe	
2005/0204385 A1	9/2005	Sull et al.	
2005/0223087 A1	10/2005	Van Der Stok	
2005/0251832 A1	11/2005	Chiueh	
2005/0254508 A1	11/2005	Aksu et al.	
2005/0262257 A1	11/2005	Major et al.	
2006/0010003 A1	1/2006	Kruse	
2006/0047779 A1	3/2006	Deshpande	
2006/0059223 A1	3/2006	Klemets et al.	
2006/0080718 A1	4/2006	Gray et al.	
2006/0130118 A1	6/2006	Damm	
2006/0133809 A1	6/2006	Chow et al.	
2006/0165166 A1	7/2006	Chou et al.	
2006/0168290 A1	7/2006	Doron	
2006/0168295 A1	7/2006	Batterberry et al.	
2006/0184688 A1	8/2006	Ganguly et al.	
2006/0206246 A1	9/2006	Walker	
2006/0218264 A1	9/2006	Ogawa et al.	
2006/0236219 A1	10/2006	Grigorovitch et al.	
2006/0242315 A1	10/2006	Nichols	
2006/0270404 A1	11/2006	Tuohino et al.	
2006/0277564 A1	12/2006	Jarman	
2006/0282540 A1	12/2006	Tanimoto	
2006/0288099 A1	12/2006	Jefferson et al.	
2007/0024705 A1	2/2007	Richter et al.	
2007/0030833 A1	2/2007	Pirzada et al.	
2007/0037599 A1	2/2007	Tillet et al.	
2007/0067480 A1	3/2007	Beek et al.	
2007/0078768 A1	4/2007	Dawson	
2007/0079325 A1	4/2007	de Heer	
2007/0094405 A1	4/2007	Zhang	
2007/0204310 A1	8/2007	Hua et al.	
2007/0280255 A1	12/2007	Tsang et al.	
2008/0028428 A1	1/2008	Jeong et al.	
2008/0037527 A1	2/2008	Chan et al.	
2008/0046939 A1	2/2008	Lu et al.	
2008/0056373 A1	3/2008	Newlin et al.	
2008/0060029 A1	3/2008	Park et al.	
2008/0091838 A1	4/2008	Miceli	
2008/0133766 A1	6/2008	Luo	
2008/0162713 A1	7/2008	Bowra et al.	
2008/0195744 A1	8/2008	Bowra et al.	
2008/0195745 A1	8/2008	Bowra et al.	
2008/0205291 A1	8/2008	Li et al.	
2008/0219151 A1	9/2008	Ma et al.	
2008/0263180 A1	10/2008	Hurst et al.	
2008/0281803 A1	11/2008	Gentric	
2009/0006538 A1	1/2009	Risney, Jr. et al.	
2009/0049186 A1	2/2009	Agnihotri et al.	
2009/0055417 A1	2/2009	Hannuksela	
2009/0055471 A1	2/2009	Kozat et al.	
2009/0055547 A1	2/2009	Hudson et al.	
2009/0132599 A1	5/2009	Soroushian et al.	
2009/0132721 A1	5/2009	Soroushian et al.	
2009/0210549 A1	8/2009	Hudson et al.	
2010/0098103 A1	4/2010	Xiong et al.	
2010/0158101 A1	6/2010	Wu et al.	
2010/0262711 A1	10/2010	Bouazizi	
2014/0207966 A1	7/2014	Hurst et al.	
2015/0058496 A1	2/2015	Hurst et al.	

FOREIGN PATENT DOCUMENTS

EP	0 919 952	A1	6/1999
EP	1202487	A2	10/2001
EP	1395014	A1	8/2002
EP	1298931	A2	2/2003
EP	1298931	A2	4/2003
EP	1 641 271	A2	3/2006
EP	1 670 256	A2	6/2006
EP	1 777 969	A1	4/2007
GB	2367219	A	9/2000
JP	2000201343	A	7/2000
JP	200192752		4/2001
JP	2004054930		2/2004
JP	2011004225	A	1/2011

(56) **References Cited**

FOREIGN PATENT DOCUMENTS

WO	WO 00/67469		11/2000
WO	0167264	A1	9/2001
WO	03003760	A2	1/2003
WO	03009581	A1	1/2003
WO	03027876	A1	4/2003
WO	2004025405	A2	3/2004
WO	2004036824	A1	4/2004
WO	2006010113	A2	1/2006

OTHER PUBLICATIONS

U.S. Patent and Trademark Office, Non-Final Office Action, dated Oct. 24, 2014 for U.S. Appl. No. 14/222,245.
USPTO, Notice of Allowance and Fee(s) Due for U.S. Appl. No. 14/106,051 mailed Feb. 24, 2015.
USPTO, Final Office Action for U.S. Appl. No. 14/222,245 mailed Mar. 18, 2015.
Canadian Intellectual Property Office, Office Action, dated Sep. 10, 2014 for Canadian Application No. 2564861.
USPTO "International Search Report" mailed Dec. 12, 2008; International Appln. No. PCT/US2008/061035, filed Apr. 21, 2008.
Australian Government "Examiner's First Report" dated Oct. 17, 2011; Australian Patent Appln. No. 2011213730.
Korean Intellectual Property Office "Official Notice of Preliminary Rejection" issued Jul. 28, 2011; Korean Patent Appln. No. 10-2006-7025274.
Japan Patent Office "Notice of Rejection Ground" mailed Apr. 26, 2011; Japanese Patent Appln. No. 2007-511070.
Fujisawa, Hiroshi et al. "Implementation of Efficient Access Mechanism for Multiple Mirror-Servers" IPSJ SIG Technical Report, vol. 2004, No. 9, Jan. 30, 2004, Information Processing Society of Japan, pp. 37-42.
Liu, Jiangchuan et al. "Opportunities and Challenged of Peer-to-Peer Internet Video Broadcast," School of Computing Science, Simon Fraser University, British Columbia, Canada.
USPTO International Searching Authority "International Search Report and Written Opinion," mailed Nov. 5, 2008; International Appln. No. PCT/US2008/009281, filed Aug. 1, 2008.
Zhang, Xinyan et al. "CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming" IEEE 2005.
Guo, Yang "DirectStream: A Directory-Based Peer-To-Peer Video Streaming Service" LexisNexis, Elsevier B.V. 2007.
Liu, Jiangchuan et al. "Adaptive Video Multicast Over the Internet" IEEE Computer Society, 2003.
Rejaie, Reza et al. "Architectural Considerations for Playback of Quality Adaptive Video Over the Internet" University of Southern California, Information Sciences Institute, 1998.)
Roy, Sumit et al. "A System Architecture for Managing Mobile Streaming Media Services" Streaming Media Systems Group, Hewlett-Packard Laboratories, 2003.
Xu, Dongyan et al. "On Peer-to-Peer Media Streaming" Department of Computer Sciences, Purdue University, 2002.
Kozamernik, Franc "Media Streaming Over the Internet—An Over of Delivery Technologies" EBU Technical Review, Oct. 2002.
Lienhart, Rainer et al. "Challenges in Distributed Video Management and Delivery" Intel Corporation, EECS Dept., UC Berkeley, 2000-2002.
Japan Patent Office "Final Office Action" mailed Feb. 28, 2012 in Patent Application No. 2007-511070 filed on Oct. 26, 2006.
Japan Patent Office "Interrogation" mailed Nov. 6, 2012 in Patent Application No. 2007-511070 filed on Oct. 26, 2006.
Canadian Intellectual Property Office "Office Action" mailed Sep. 9, 2013 in Patent Application No. 2,564,861 filed on Oct. 30, 2006.
USPTO "Office Action" mailed Sep. 13, 2013 in U.S. Appl. No. 13/757,571, filed Feb. 1, 2013.
USPTO "Notice of Allowance" mailed Jun. 24, 2014 in U.S. Appl. No. 13/757,571, filed Feb. 1, 2013.
European Patent Office "Extended Search Report" dated Jul. 10, 2014 in Patent Application No. 12154559.4 filed on Sep. 20, 2002.
Nguyen, Thinh, "Multiple Sender Distributed Video Streaming" in IEEE Transactions on Multimedia, vol. 6, No. 2, Published Apr. 2, 2004.
Weblio, The Meaning of Performance Factor—English-Japanese Weblio Dictionary, [online], Feb. 24, 2012; retrieved from the internet—URL:http://ejje, weblio.jp/content/performance+factor.
Masato Tsuru et al., Recent Evolution of the Internet Measurement and Inference Techniques, IEICE Technical Report, vol. 103, No. 123 (IN2003-16 to 23), IEICE, Jun. 12, 2003, pp. 37 to 42, ISSN: 0913-05685.
Takeshi Yoshimura et al., Mobile Streaming Media CDN Enabled by Dynamic SMIL, WWW2002, May 7-11, 2002; retrieved from the Internet at http://www2002.org/CDROM/refereed/515/.
Canadian Intellectual Property Office, Office Action, mailed Oct. 15, 2012 for Patent Application No. 2,564,861.
Clement, B., Move Networks Closes $11.3 Million on First Round VC Funding, Page One PR, Move Networks, Inc. Press Releases, Feb. 7, 2007, http://www.move.tv/press/press20070201.html.
Move Networks, Inc., The Next Generation Video Publishing System, Apr. 11, 2007; http://www.movenetworks.com/wp-content/uploads/move-networks-publishing-system.pdf.
U.S. Patent and Trademark Office, Non-Final Office Action, dated Aug. 7, 2014 for U.S. Appl. No. 14/106,051.
Final Office Action for U.S. Appl. No. 11/673,483, Feb. 4, 2010, 21 pages.
Advisory Action for U.S. Appl. No. 11/673,483, Apr. 9, 2010, 3 pages.
Advisory Action for U.S. Appl. No. 11/673,483, May 26, 2010, 3 pages.
Notice of Allowance for U.S. Appl. No. 11/673,483, Aug. 5, 2010, 7 pages.
Wicker, Stephen B., "Error Control Systems for Digital Communication and Storage", Prentice-Hall, Inc., New Jersey, USA, 1995 (Book: see NPL's Parts 1-6).
PCT Notification of Transmittal of the International Search Report and Written Opinion of the International Searching Authority, for PCT/US05/15091, Oct. 29, 2007, 8 pages.
PCT Notification of Transmittal of International Preliminary Report on Patentability, for PCT/US05/15091, Oct. 29, 2007, 6 pages.
Office Action for U.S. Appl. No. 11/673,483, Jul. 9, 2009, 14 pages.
Office Action for U.S. Appl. No. 11/673,483, Feb. 3, 2009, 9 pages.
Albanese, Andres, et al. "Priority Encoding Transmission", TR-94-039, Aug. 1994, 36 pages, International Computer Science Institute, Berkeley, California.
Puri, Rohit, et al. "Multiple Description Source Coding Using Forward Error Correction Codes", Oct. 1999, 5 pages, Department of Electrical Engineering and Computer Science, University of California, Berkeley, California.
Goyal, Vivek K., "Multiple Description Coding: Compression Meets the Network", Sep. 2001, pp. 74-93, IEEE Signal Processing Magazine.
Supplemental European Search Report, Sep. 30, 2008, (3 pages).
Pathan, Al-Mukaddim, et al., "A Taxonomy and Survey of Content Delivery Networks", Australia, Feb. 2007. Available at http://gridbus.org/reports/CDN-Taxonomy.pdf.
On2 Technologies, Inc., "TrueMotion VP7 Video Codec", White Paper, Document Version 1.0, Jan. 10, 2005, (13 pages).
USPTO, Office Action for U.S. Appl. No. 14/531,804, mailed May 11, 2015.

* cited by examiner

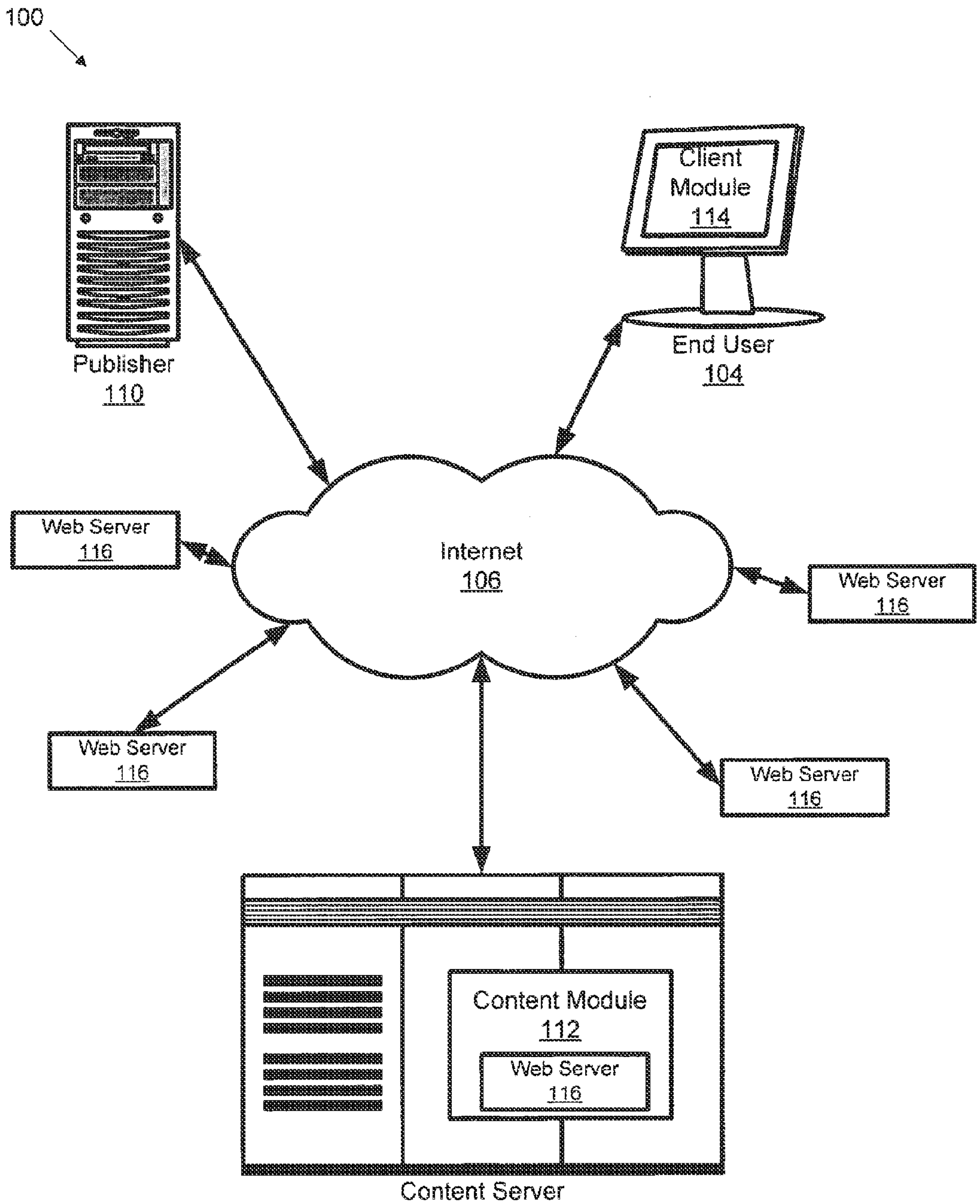

100
Client Module 114
Publisher 110
End User 104
Web Server 116
Internet 106
Web Server 116
Web Server 116
Web Server 116
Content Module 112
Web Server 116
Content Server

FIG. 1

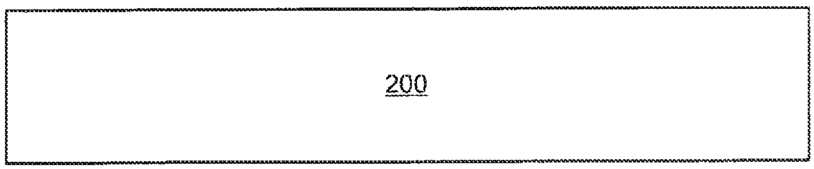


FIG. 2a

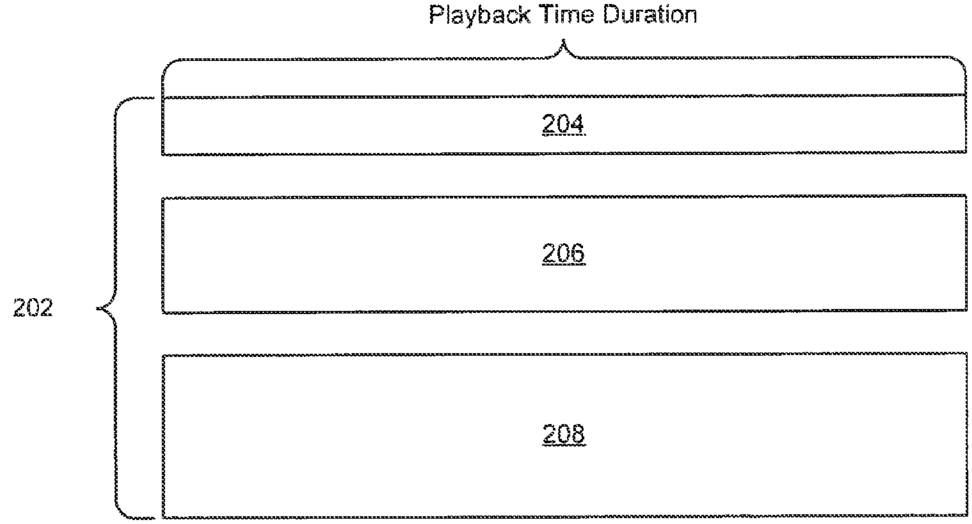


FIG. 2b

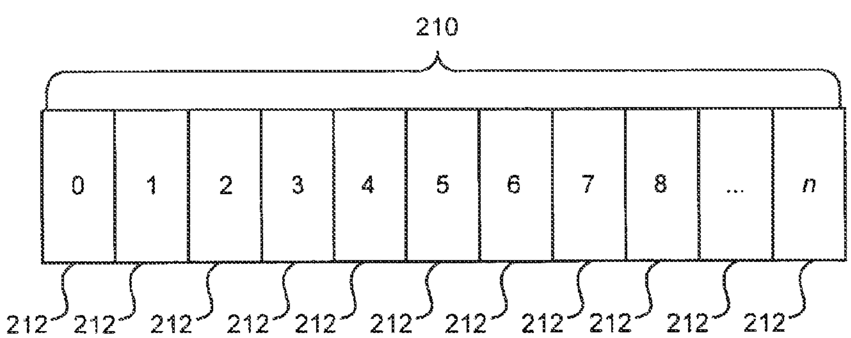


FIG. 2c

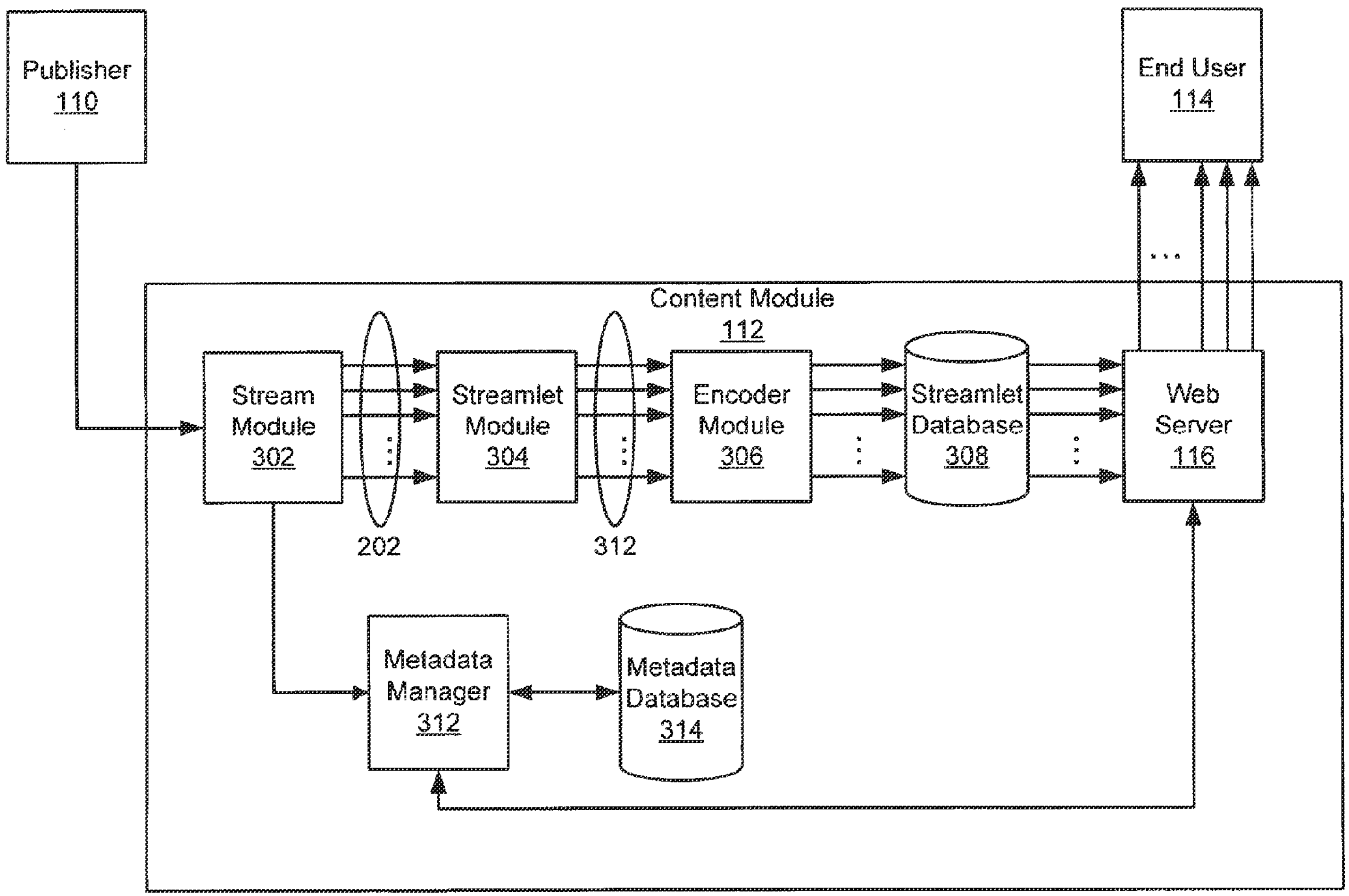

Publisher
110
End User
114
Content Module
112
Stream
Module
302
202
Streamlet
Module
304
312
Encoder
Module
306
Streamlet
Database
308
Web
Server
116
Metadata
Manager
312
Metadata
Database
314

FIG. 3

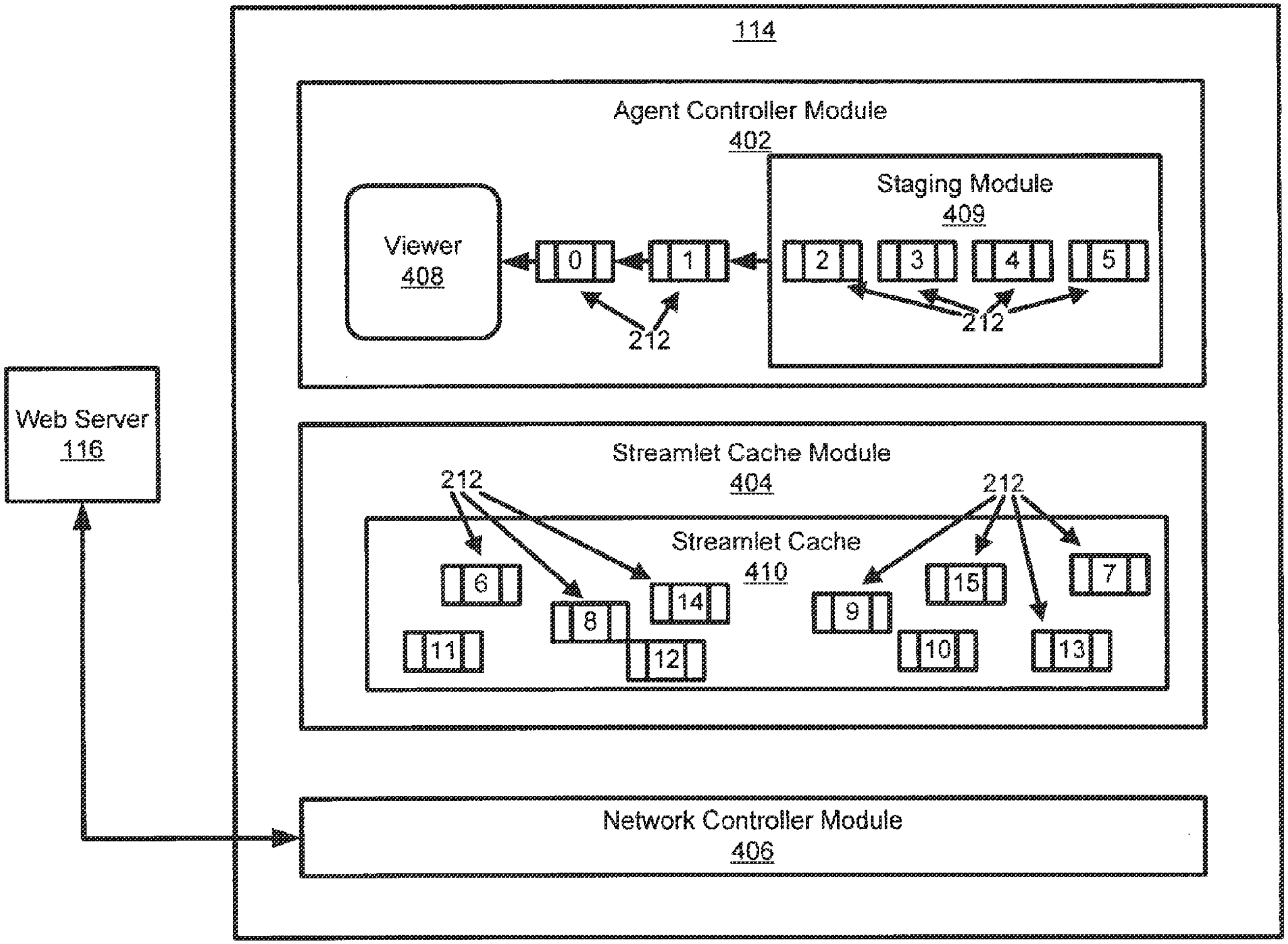

114
Agent Controller Module
402
Viewer
408
0
1
212
Staging Module
409
2
3
4
5
212
Web Server
116
Streamlet Cache Module
404
212
212
Streamlet Cache
410
6
14
8
11
12
9
15
7
10
13
Network Controller Module
406

FIG. 4

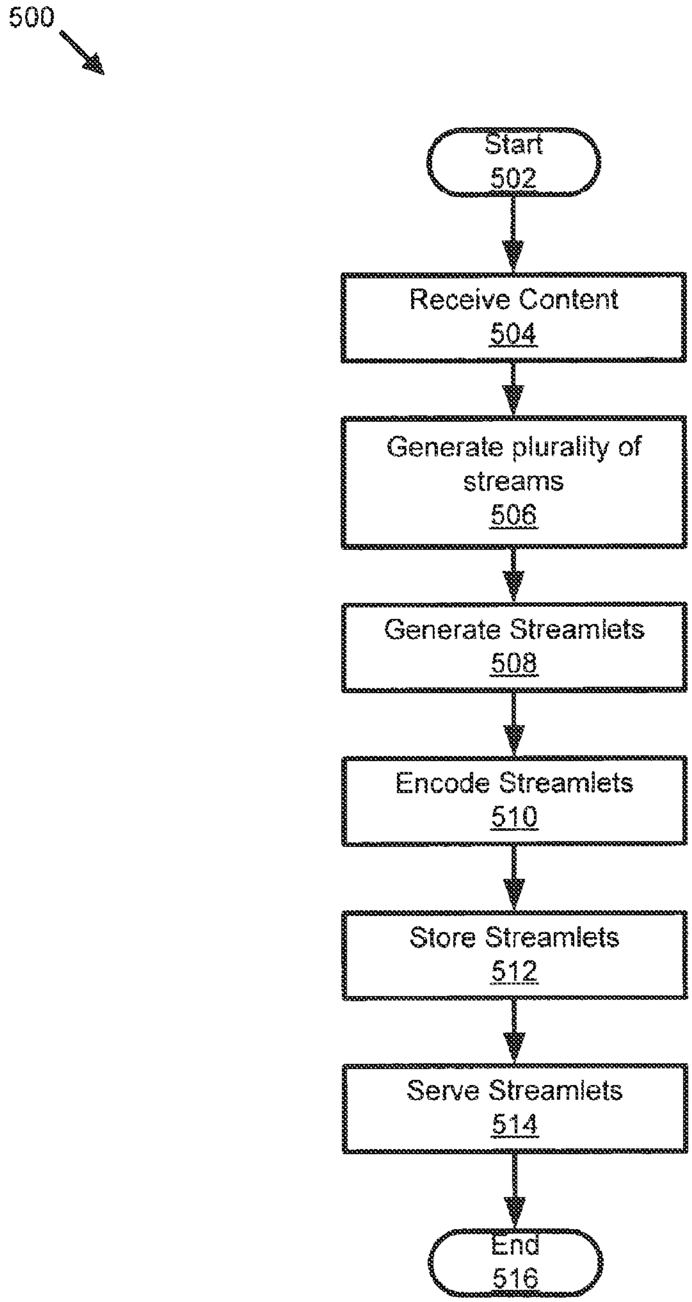
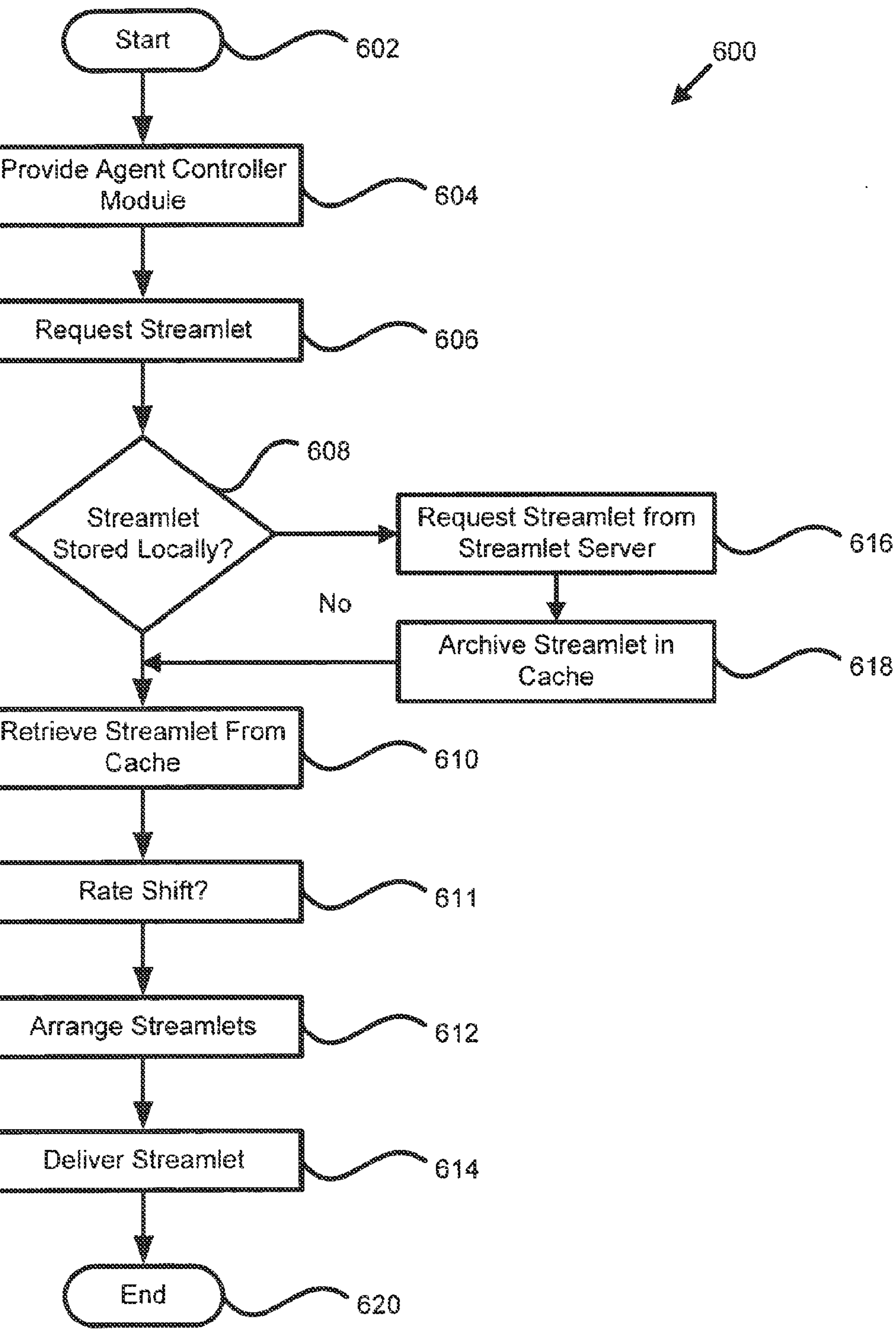
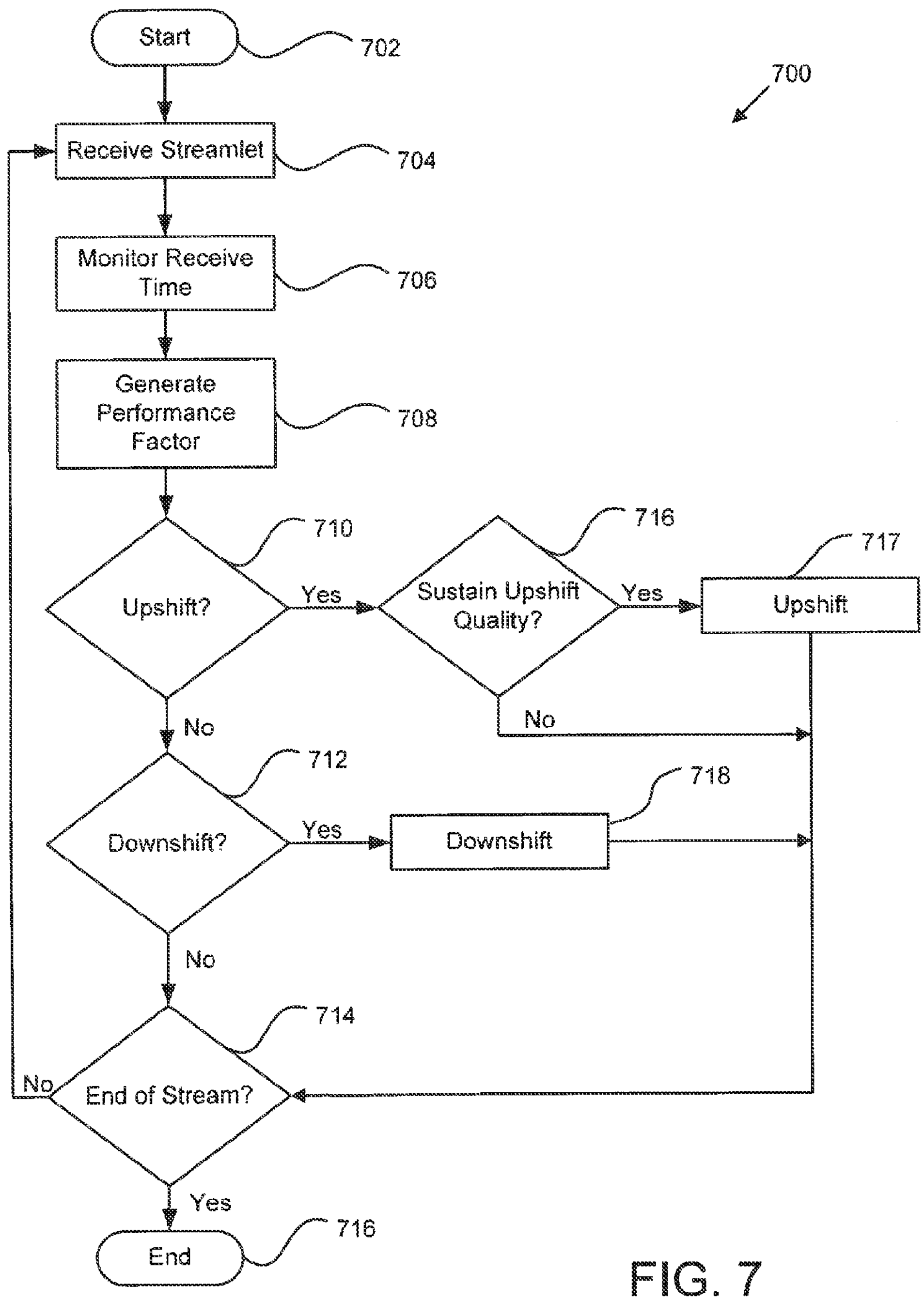

500
Start
502
Receive Content
504
Generate plurality of streams
506
Generate Streamlets
508
Encode Streamlets
510
Store Streamlets
512
Serve Streamlets
514
End
516

FIG. 5



Start
602
600
Provide Agent Controller Module
604
Request Streamlet
606
608
Streamlet Stored Locally?
Request Streamlet from Streamlet Server
616
No
Archive Streamlet in Cache
618
Retrieve Streamlet From Cache
610
Rate Shift?
611
Arrange Streamlets
612
Deliver Streamlet
614
End
620

FIG. 6



700
Start
702
Receive Streamlet
704
Monitor Receive Time
706
Generate Performance Factor
708
Upshift?
710
Yes
Sustain Upshift Quality?
716
Yes
Upshift
717
No
No
Downshift?
712
Yes
Downshift
718
No
End of Stream?
714
No
Yes
End
716

FIG. 7

APPARATUS, SYSTEM, AND METHOD FOR ADAPTIVE-RATE SHIFTING OF STREAMING CONTENT

CROSS-REFERENCES TO RELATED APPLICATIONS

This application claims benefit of U.S. Provisional Patent Application No. 60/566,831 entitled "APPARATUS, SYSTEM, AND METHOD FOR DYNAMIC RATE SHIFTING OF STREAMING CONTENT" and filed on Apr. 30, 2004 for R. Drew Major and Mark B. Hurst, which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to video streaming over packet switched networks such as the Internet, and more particularly relates to adaptive-rate shifting of streaming content over such networks.

2. Description of the Related Art

The Internet is last becoming a preferred method for distributing media files to end users. It is currently possible to download music or video to computers, cell phones, or practically any network capable device. Many portable media players are equipped with network connections and enabled to play music or videos. The music or video files (hereinafter "media files") can be stored locally on the media player or computer, or streamed or downloaded from a server.

"Streaming media" refers to technology that delivers content at a rate sufficient for presenting the media to a user in real time as the data is received. The data may be stored in memory temporarily until played and then subsequently deleted. The user has the immediate satisfaction of viewing the requested content without wading for the media file to completely download. Unfortunately, the audio/video quality that can be received for real time presentation is constrained by the available bandwidth of the user's network connection. Streaming may be used to deliver content on demand (previously recorded) or from live broadcasts.

Alternatively, media files may be downloaded and stored on persistent storage devices, such as hard drives or optical storage, for later presentation. Downloading complete media files can take large amounts of time depending on the network connection. Once downloaded, however, the content can be viewed repeatedly anytime or anywhere. Media files prepared for downloading usually are encoded with a higher quality audio/video than can be delivered in real time. Users generally dislike this option, as they tend to want to see or hear the media file instantaneously.

Streaming offers the advantage of immediate access to the content but currently sacrifices quality compared with downloading a file of the same content. Streaming also provides the opportunity for a user to select different content for viewing on an ad hoc basis, while downloading is by definition restricted to receiving a specific content selection in its entirety or not at all. Downloading also supports rewind, fast forward, and direct seek operations, while streaming is unable to fully support these functions. Streaming is also vulnerable to network failures or congestion.

Another technology, known as "progressive downloads," attempts to combine the strengths of the above two technologies. When a progressive download is initiated, the media file download begins, and the media player waits to begin playback until there is enough of the file downloaded that playback can begin with the hope that the remainder of the file will be completely downloaded before playback "catches up." This waiting period before playback can be substantial depending on network conditions, and therefore is not a complete or fully acceptable solution to the problem of media presentation over a network.

Generally, three basic challenges exist with regard to data transport streaming over a network such as the Internet that has a varying amount of data loss. The first challenge is reliability. Most streaming solutions use a TCP connection, or "virtual circuit," for transmitting data. A TCP connection provides a guaranteed delivery mechanism so that data sent from one endpoint will be delivered to the destination, even if portions are lost and retransmitted. A break in the continuity of a TCP connection can have serious consequences when the data must be delivered in real-time. When a network adapter detects delays or losses in a TCP connection, the adapter "backs off" from transmission attempts for a moment and then slowly resumes the original transmission pace. This behavior is an attempt to alleviate the perceived congestion. Such a slowdown is detrimental, to the viewing or listening experience of the user and therefore is not acceptable.

The second challenge to data transport Is efficiency. Efficiency refers to how well the user's available bandwidth is used for delivery of the content stream. This measure is directly related to the reliability of the TCP connection. When the TCP connection is suffering reliability problems, a loss of bandwidth utilization results. The measure of efficiency sometimes varies suddenly, and can greatly impact the viewing experience.

The third challenge is latency. Latency is the time measure form the client's point-of-view of the interval between when a request is issued and the response data begins to arrive. This value is affected by the network connection's reliability and efficiency, and the processing time required by the origin to prepare the response. A busy or overloaded server, for example, will take more time to process a request. As well as affecting the start time of a particular request, latency has a significant impact on the network throughput of TCP.

From the foregoing discussion, it should be apparent that a need exists for an apparatus, system, and method that alleviate the problems of reliability, efficiency, and latency. Additionally, such an apparatus, system, and method would offer instantaneous viewing along with the ability to fast forward, rewind, direct seek, and browse multiple streams. Beneficially, such an apparatus, system, and method would utilize multiple connections between a source and destination, requesting varying bitrate streams depending upon network conditions.

SUMMARY OF THE INVENTION

The present invention has been developed in response to the present state of the art, and in particular, in response to the problems and needs in the art that have not yet been fully solved by currently available content streaming systems. Accordingly, the present invention has been developed to provide an apparatus, system, and method for adaptive-rate content streaming that overcome many or all of the above-discussed shortcomings in the art.

The apparatus for adaptive-rate content streaming is provided with a logic unit containing a plurality of modules configured to functionally execute the necessary steps. These modules in the described embodiments include an agent controller module configured to simultaneously request a plurality of streamlets, the agent controller module further configured to continuously monitor streamlet requests and subsequent responses, and accordingly request higher or

lower quality streamlets, and a staging module configured to stage the streamlets and arrange the streamlets for playback on a content player.

The apparatus is further configured, in one embodiment, to establish multiple Transmission Control Protocol (TCP) connections with a content server, and request streamlets of varying bitrates. Each streamlet may further comprise a portion of a content file. Additionally, the agent controller module may be configured to generate a performance factor according to responses from streamlet requests.

In a further embodiment, the agent controller module is configured to upshift to a higher quality streamlet when the performance factor is greater than a threshold, and the agent controller module determines the higher quality playback can be sustained according to a combination of factors. The factors may include an amount of contiguously available streamlets stored in the staging module, a minimum safety margin, and a current read, ahead margin.

The agent controller module may be configured to downshift to a lower quality streamlet when the performance factor is less than a second threshold. Also, the agent controller module is further configured to anticipate streamlet requests and pre-request streamlets to enable fast-forward, skip randomly, and rewind functionality. In one embodiment, the agent controller module is configured to initially request low quality streamlets to enable instant playback of the content file, and subsequent upshifting according to the performance factor.

A system of the present invention is also presented to adaptive-rate content streaming. In particular, the system, in one embodiment, includes a data communications network, and a content server coupled to the data communications network and having a content module configured to process content and generate a plurality of high and low quality streams. In one embodiment, each of the high and low quality streams may include a plurality of streamlets.

In a further embodiment, the system also includes an agent controller module configured to simultaneously request a plurality of streamlets, the agent controller module further configured to continuously monitor streamlet requests and subsequent responses, and accordingly request higher or lower quality streamlets, and a staging module configured to stage the streamlets and arrange the streamlets for playback on a content player.

A method of the present invention is also presented for adaptive-rate content streaming. The method in the disclosed embodiments substantially includes the steps necessary to carry out the functions presented above with respect to the operation of the described apparatus and system. In one embodiment, the method includes simultaneously requesting a plurality of streamlets, continuously monitoring streamlet requests and subsequent responses, and accordingly requesting higher or lower quality streamlets, and staging the streamlets and arranging the streamlets for playback on a content player.

In a further embodiment, the method may include establishing multiple Transmission Control Protocol (TCP) connections with a content server, and requesting streamlets of varying nitrates. Also, the method may include generating & performance factor according to responses from streamlet requests, upshifting to a higher quality streamlet when the performance factor is greater than a threshold, and determining if the higher quality playback can be sustained. Furthermore, the method may include downshifting to a lower quality streamlet when the performance factor is less than a second threshold.

In one embodiment, the method includes anticipating streamlet requests and pre-requesting streamlets to enable fast-forward, skip randomly, and rewind functionality. The method may also comprise initially requesting low quality streamlets to enable instant playback of a content file, and subsequent upshifting according to the performance factor.

Reference throughout this specification to features, advantages, or similar language does not imply that all of the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussion of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

Furthermore, the described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize that the invention may be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

These features and advantages of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the advantages of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings, in which;

FIG. **1** is a schematic block diagram illustrating one embodiment of a system for adaptive rate shifting of streaming content in accordance with the present invention;

FIG. **2*a*** is a schematic block diagram graphically illustrating one embodiment of a content file in accordance with the present invention:

FIG. **2*b*** is a schematic block diagram illustrating one embodiment of a plurality of streams having varying degrees of quality and bandwidth in accordance with the present invention;

FIG. **2*c*** is a schematic block diagram illustrating one embodiment of a stream divided into a plurality of streamlets in accordance with the present invention;

FIG. **3** is a schematic block diagram illustrating one embodiment of a content module in accordance with die present invention;

FIG. **4** is a schematic block diagram graphically illustrating one embodiment of a client module in accordance with the present invention;

FIG. **5** is a schematic flow chart diagram illustrating one embodiment of a method for processing content in accordance with the present invention;

FIG. **6** is a schematic flow chart diagram illustrating one embodiment of a method for playback of a plurality of streamlets in accordance with the present invention; and

FIG. **7** is a schematic flow chart diagram illustrating one embodiment of a method for requesting streamlets within an adaptive-rate content streaming environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Many of the functional units described in this specification have been labeled as modules, in order to more particularly emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom VLSI circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices or the like.

Modules may also be implemented in software for execution by various types of processors. An identified module of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions which may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may comprise disparate instructions stored in different locations which, when joined logically together, comprise the module and achieve the stated purpose for the module.

Indeed, a module of executable code may be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several, memory devices. Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

Reference throughout this specification to "one embodiment," "an embodiment," or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in one embodiment," "in an embodiment," and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

Reference to a signal hearing medium may take any form capable of generating a signal, causing a signal to be generated, or causing execution of a program of machine-readable instructions on a digital processing apparatus. A signal bearing medium may be embodied by a transmission line, a compact disk, digital-video disk, a magnetic tape, a Bernoulli drive, a magnetic disk, a punch card, flash memory, integrated circuits, or other digital processing apparatus memory device.

Furthermore, the described features, structures, or characteristics of the invention may be combined in any suitable manner in one or more embodiments. In the following description, numerous specific details are provided, such as examples of programming, software modules, user selections, network transactions, database queries, database structures, hardware modules, hardware circuits, hardware chips, etc., to provide a thorough understanding of embodiments of the invention. One skilled in the relevant art will recognize, however, that the invention may be practiced without one or more of the specific details, or with other methods, components, materials, and so forth. In other instances, well-known

structures, materials, or operations are not shown or described in detail to avoid obscuring aspects of the invention.

FIG. **1** is a schematic block diagram illustrating one embodiment of a system **100** for dynamic rate shifting of streaming content in accordance with the present invention. In one embodiment, the system **100** comprises a content server **102** and an end user **104**. The content server **102** and the end user station **104** may be coupled by a data communications network. The data communications network may include the Internet **106** and connections **108** to the Internet **106**. Alternatively, the content server **102** and the end user **104** may be located on a common local area network, wireless area network, cellular network, virtual local area network, or the like. The end user station **104** may comprise a personal computer (PC), an entertainment system configured to communicate over a network, or a portable electronic device configured to present content.

In the depicted embodiment, the system **100** also includes a publisher **110**, and a web server **116**. The publisher **110** may be a creator or distributor of content. For example, it the content to be streamed were a broadcast of a television program, the publisher may be a television or cable network channel such as NBC®, or MTV®. Content may be transferred over the internet **106** to the content server **102**, where the content is received by a content module **112**. The content module **112** may fee configured to receive, process, and store content. In one embodiment, processed content is accessed by a client module **114** configured to play the content on the end user station **104**. In a further embodiment, the client module **114** is configured to receive different portions of a content stream from a plurality of locations simultaneously. For example, the client module **114** may request and receive content horn any of the plurality of web servers **116**.

FIG. **2***a* is a schematic block diagram graphically illustrating one embodiment of a content file **200**. In one embodiment, the content file **200** is distributed by the publisher **110**. The content file **200** may comprise a television broadcast, sports event, movie, music, concert, etc. The content file **200** may also be live or archived content. The content file **200** may comprise uncompressed video and audio, or alternatively, video or audio. Additionally, the content file **200** may be compressed. Examples of a compressed content file **200** include, but are not limited to, DivX®, Windows Media Video 98®, Quicklime 6.5 Sorenson 3®, or Quicklime 6.5/MPEG-4® encoded content.

FIG. **2***b* is a schematic block diagram illustrating one embodiment of a plurality of streams **202** having varying degrees of quality and bandwidth. In one embodiment, the plurality of streams **202** comprises a low quality stream **204**, a medium quality stream **206**, and a high quality stream **208**. Each of the streams **204**, **206**, **208** is a copy of the content file **200** encoded and compressed to varying bit rates. For example, the low quality stream **204** may be encoded and compressed to a bit rate of 100 kilobits per second (kbps), the medium quality stream **206** may be encoded and compressed to a bit rate of 200 kbps, and the high quality stream **208** may be encoded and compressed to 600 kbps.

FIG. **2***c* is a schematic block diagram illustrating one embodiment of a stream **210** divided into a plurality of streamlets **212**. As used herein, streamlet refers to any sized portion of the content file **200**. Each streamlet **212** may comprise a portion of the content contained in stream **210**, encapsulated as an independent media object. The content in a streamlet **212** may have a unique time index in relation to the beginning of the content contained in stream **210**. In one embodiment, the content contained in each streamlet **212** has a duration of two seconds. For example, streamlet 0 may have

a time index of 00:00 representing the beginning of content playback, and streamlet 1 may have a time index of 00:02, and so on. Alternatively, the time duration of the streamlets **212** may be any duration smaller than the entire playback duration of the content in stream **210**. In a further embodiment, the streamlets **212** may be divided according to file size instead of a time index.

FIG. **3** is a schematic block diagram illustrating in greater detail one embodiment of the content module **112** in accordance with the present invention. The content module **112** may comprise a stream module **302**, a streamlet module **304**, an encoder module **306**, a streamlet database **308**, and the web server **116**. In one embodiment, the stream module **302** is configured to receive the content file **200** from the publisher **110** and generate the plurality of streams **202** of varying qualities. The original content file **200** from the publisher may be digital in form and may comprise content having a high bit rate such as, for example, 2 mbps. The content may be transferred from the publisher **110** to the content module **112** over the Internet **106**. Such transfers of data are well known in the art and do not require further discussion herein. Alternatively, the content may comprise a captured broadcast.

In the depicted embodiment, the plurality of streams **202** may comprise the low quality stream **204**, the medium quality stream **206**, and the high quality stream **208**. Alternatively, the plurality of streams **202** may comprise any number of streams deemed necessary to accommodate end user bandwidth. The streamlet module **304** may be configured to receive the plurality of streams **202** from the stream module and generate a plurality of streams **312**, each stream comprising a plurality of streamlets **212**. As described with reference to FIG. **2***c*, each streamlet **212** may comprise a pre-defined portion of the stream. The encoder module **306** is configured to encode each streamlet from the plurality of streams **312** and store the streamlets in the streamlet database **308**. The encoding module **306** may utilize encoding schemes such as DivX®, Windows Media Video 9®, Quicklime 6.5 Sorenson 3®, or Quicklime 6.5/MPEG-4®. Alternatively, a custom encoding scheme may be employed.

The content module **112** may also include a metadata module **312** and a metadata database **314**. In one embodiment, metadata comprises static searchable content information. For example, metadata includes, but is not limited to, air date of the content, title, actresses, actors, length, and episode name. Metadata is generated by the publisher **110**, and may be configured to define an end user environment. In one embodiment, the publisher **100** may define an end user navigational environment for the content including menus, thumbnails, sidebars, advertising, etc. Additionally, the publisher **110** may define functions such as fast forward, rewind, cause, and play that may be used with the content file **200**. The metadata module **312** is configured to receive the metadata from the publisher **110** and store the metadata in the metadata database **314**. In a further embodiment, the metadata module **312** is configured to interface with the client module **114**, allowing the client module **114** to search for content based upon at least one of a plurality of metadata criteria. Additionally, metadata may be generated by the content module **112** through automated process(es) or manual definition.

Once the streamlets **212** have been received and processed, the client module **114** may request streamlets **212** using HTTP from the web server **116**. Such use of client side initiated requests requires no additional configuration of firewalls. Additionally, since the client module **114** initiates the request, the web server **116** is only required to retrieve and serve the requested streamlet. In a further embodiment, the client module **114** may be configured to retrieve streamlets

212 from a plurality of web servers **310**. Each web server **116** may be located in various locations across the Internet **106**. The streamlets **212** are essentially static files. As such, no specialized media server or server-side intelligence is required for a client module **114** to retrieve streamlets **212**. Streamlets **212** may be served by the web server **116** or cached by cache servers of Internet Service Providers (ISPs), or any other network infrastructure operators, and served by the cache server. Use of cache servers is well known to those skilled in the art, and will not be discussed further herein. Thus, a highly scalable solution is provided that is not hindered by massive amounts of client module **114** requests to the web server **116** at any specific location.

FIG. **4** is a schematic block diagram graphically illustrating one embodiment of a client module **114** in accordance with the present invention. The client module **114** may comprise an agent controller module **402**, a streamlet cache module **404**, and a network controller module **406**. In one embodiment, the agent controller module **402** is configured to interface with a viewer **408**, and transmit streamlets **212** to the viewer **408**, In a further embodiment, the client module **114** may comprise a plurality of agent controller modules **402**. Each agent controller module **402** may be configured to interface with one viewer **408**. Alternatively, the agent controller module **402** may be configured to interface with a plurality of viewers **408**. The viewer **408** may be a media player (not shown) operating on a PC or handheld electronic device.

The agent controller module **402** is configured to select a quality level of streamlets to transmit to the viewer **408**. The agent controller module **402** requests lower or higher quality streams based upon continuous observation, of time intervals between successive receive times of each requested streamlet. The method of requesting higher or lower quality streams will be discussed in greater detail below with reference to FIG. **7**.

The agent controller module **402** may be configured to receive user commands from the viewer **408**. Such commands may include play, fast forward, rewind, pause, and stop. In one embodiment, the agent controller module **402** requests streamlets **212** from the streamlet cache module **404** and arranges the received streamlets **212** in a staging module **409**. The staging module **409** may fee configured to arrange the streamlets **212** in order of ascending playback time. In the depleted embodiment, the streamlets **212** are numbered 0, 1, 2, 3, 4, etc. However, each streamlet **212** may be identified with a unique filename.

Additionally, the agent controller module **402** may be configured to anticipate streamlet **212** requests and pre-request streamlets **212**. By pro-requesting streamlets **212**, the user may fast-forward, skip randomly, or rewind through the content and experience no buffering delay. In a further embodiment, the agent controller module **402** may request the streamlets **212** that correspond to time index intervals of 30 seconds within the total play time of the content. Alternatively, the agent controller module **402** may request streamlets at any interval less than the length of the time index. This enables a "fast-start" capability with no buffering wait when starting or fast-forwarding through content file **200**. In a further embodiment, the agent controller module **402** may be configured to pre-request streamlets **212** corresponding to specified Index points within the content or within other content in anticipation of the end user **104** selecting new content to view.

In one embodiment, the streamlet cache module **404** is configured to receive streamlet **212** requests from the agent controller module **402**. Upon receiving a request, the streamlet cache module **404** first checks a streamlet cache **410** to verify if the streamlet **212** is present. In a further embodiment,

the streamlet cache module **404** handles streamlet **212** requests from a plurality of agent controller modules **402**. Alternatively, a streamlet cache module **404** may be provided for each agent controller module **402**. If the requested streamlet **212** is not present m the streamlet cache **410**, the request is passed to the network controller module **406**. In order to enable last forward and rewind capabilities, the streamlet cache module **404** is configured to store the plurality of streamlets **212** in the streamlet cache **410** for a specified time period after the streamlet **212** has been viewed. However, once the streamlets **212** have been deleted, they may be requested again from the web server **116**.

The network controller module **406** may be configured to receive streamlet requests from the streamlet cache module **404** and open a connection to the web server **116** or other remote streamlet **212** database (not shown). In one embodiment, the network controller module **406** opens a TCP/IP connection to the web server **116** and generates a standard HTTP GET request for the requested streamlet **212**. Upon receiving the requested streamlet **212**, the network controller module **406** passes the streamlet **212** to the streamlet cache module **404** where it is stored in the streamlet cache **410**. In a further embodiment, the network controller module **406** is configured to process and request a plurality of streamlets **212** simultaneously. The network controller module **406** may also be configured to request a plurality of streamlets, where each streamlet **212** is subsequently requested in multiple parts.

In a further embodiment, streamlet requests may comprise requesting pieces of any streamlet file. Splitting the streamlet **212** into smaller pieces or portions beneficially allows for an increased efficiency potential, and also eliminates problems associated with multiple full-streamlet requests sharing the bandwidth at any given moment. This is achieved by using parallel TCP/IP connections for pieces of the streamlets **212**. Consequently, efficiency and network loss problems are overcome, and the streamlets arrive with more useful and predictable timing.

In one embodiment, the client module **114** is configured to use multiple TCP connections between the client module **114** and the web server **116** or web cache. The intervention of a cache may be transparent to the client or configured by the client as a forward cache, By requesting more than one streamlet **212** at a time in a manner referred to as "parallel retrieval," or more than one part of a streamlet **212** at a time, efficiency is raised significantly and latency is virtually eliminated. In a further embodiment, the client module allows a maximum of three outstanding streamlet **212** requests. The client module **114** may maintain additional open TCP connections as spares to be available should another connection fail. Streamlet **212** requests are rotated among all open connections to keep the TCP flow logic for any particular connection from falling into a slow-start or close mode. If the network controller module **406** has requested a streamlet **212** in multiple parts, with each part requested on mutually independent TCP/IP connections, the network controller module **406** reassembles the parts to present a complete streamlet **212** for use by all other components of the client module **114**.

When a TCP connection fails completely, a new request may be sent on a different connection for the same streamlet **212**. In a further embodiment, if a request is not being satisfied in a timely manner, a redundant request may be sent on a different connection for the same streamlet **212**, If the first, streamlet request's response arrives before the redundant request response, the redundant request can be aborted. If the redundant request response arrives before the first request response, the first request may be aborted.

Several streamlet **212** requests may be sent on a single TCP connection, and the responses are caused to flow back in matching order along the same connection. This eliminates all but the first request latency. Because multiple responses are always being transmitted, the processing latency of each new streamlet **212** response after the first is not a factor in performance. This technique is known in the industry as "pipelining." Pipelining offers efficiency in request-response processing by eliminating most of the effects of request latency. However, pipelining has serious vulnerabilities. Transmission delays affect all of the responses. If the single TCP connection fails, all of the outstanding requests and responses are lost. Pipelining causes a serial dependency between the requests.

Multiple TCP connections may be opened between the client module **114** and the web server **116** to achieve the latency-reduction efficiency benefits of pipelining while maintaining the independence of each streamlet **212** request. Several streamlet **212** requests may be sent concurrently, with each request being sent on a mutually distinct TCP connection. This technique is labeled "virtual pipelining" and is an innovation of the present invention. Multiple responses may be in transit concurrently, assuring that communication bandwidth between the client module **114** and the web server **116** is always being utilized. Virtual pipelining eliminates the vulnerabilities of traditional pipelining. A delay in or complete failure of one response does not affect the transmission of other responses because each response occupies an independent TCP connection. Any transmission bandwidth not in use by one of multiple responses (whether due to delays or TCP connection failure) may be utilized by other outstanding responses.

A single streamlet **212** request may be issued for an entire streamlet **212**, or multiple requests may be issued, each for a different part or portion of the streamlet. If the streamlet is requested in several parts, the parts may be recombined by the client module **114** streamlet.

In order to maintain a proper balance between maximized bandwidth utilization and response time, the issuance of new streamlet requests must be timed such that the web server **116** does not transmit the response before the client module **114** has fully received a response to one of the previously outstanding streamlet requests. For example, if three streamlet **212** requests are outstanding, the client module **114** should issue the next request slightly before one of the three responses is fully received and "out of the pipe." In other words, request timing is adjusted to keep three responses in transit. Sharing of bandwidth among four responses diminishes the net response time of the other three responses. The timing adjustment may be calculated dynamically by observation, and the request timing adjusted accordingly to maintain the proper balance of efficiency and response times.

The schematic flow chart diagrams that follow are generally set forth as logical flow chart diagrams. As such, the depicted order and labeled steps are indicative of one embodiment of the presented method. Other steps and methods may be conceived that are equivalent in function, logic, or effect to one or more steps, or portions thereof of the illustrated method. Additionally, the format and symbols employed are provided to explain the logical steps of the method and are understood not to limit the scope of the method. Although various arrow types and line types may be employed in the flow chart diagrams, they are understood not to limit the scope of the corresponding method. Indeed, some arrows or other connectors may be used to indicate only the logical flow of the method. For instance, an arrow may indicate a wading or monitoring period of unspecified duration between enumer-

ated steps of the depicted method. Additionally, the order in which a particular method occurs may or may not strictly adhere to the order of the corresponding steps shown.

FIG. **5** is a schematic flowchart diagram illustrating one embodiment of a method **500** for processing content in accordance with the present invention. In one embodiment the method **500** starts **502**, and the content module **112** receives **504** content from the publisher **110**. Receiving content **504** may comprise receiving **504** a digital copy of the content file **200**, or digitizing a physical copy of the content file **200**. Alternatively, receiving **504** content may comprise capturing a radio or television broadcast. Once received **504**, the stream module **302** generates **506** a plurality of streams **202**, each stream **202** having a different quality. The quality may be predefined, or automatically set according to end user bandwidth, or in response to pre-designated publisher guidelines

The streamlet module **304** receives the streams **202** and generates **508** a plurality of streamlets **212**. In one embodiment, generating **508** streamlets comprises dividing the stream **202** into a plurality of two second streamlets **212**. Alternatively, the streamlets may have any length less than or equal to the length of the stream **202**, The encoder module **306** then encodes **510** the streamlets according to a compression algorithm. In a further embodiment, the algorithm comprises a proprietary codec such as WMV9®. The encoder module **306** then stores **512** the encoded streamlets in the streamlet database **308**. Once stored **512**, the web server **116** may then serve **514** the streamlets. In one embodiment, serving **514** the streamlets comprises receiving streamlet requests from the client module **114**, retrieving the requested streamlet from the streamlet database **308**, and subsequently transmitting the streamlet to the client module **114**. The method **500** then ends **516**.

FIG. **6** is a schematic flow chart diagram illustrating one embodiment of a method **600** for viewing a plurality of streamlets in accordance with the present invention. The method **600** starts and an agent control module **402** is provided **604** and associated with a viewer **408** and provided with a staging module **409**. The agent controller module **402** then requests **606** a streamlet from the streamlet cache module **404**. Alternatively, the agent controller module **402** may simultaneously request **606** a plurality of streamlets from the streamlet cache module **404**. If the streamlet is stored **608** locally in the streamlet cache **410**, the streamlet cache module **404** retrieves **610** the streamlet and sends the streamlet to the agent controller module **402**. Upon retrieving **610** or receiving a streamlet, the agent controller module **402** makes **611** a determination of whether or not to shift, to a higher or lower quality stream **202**. This determination will be described below in greater detail with reference to FIG. **7**.

In one embodiment, the staging module **409** then arranges **612** tire streamlets into the proper order, and the agent controller module **402** delivers **614** the streamlets to the viewer **408**. In a further embodiment, delivering **614** streamlets to the end user comprises playing video and or audio streamlets on the viewer **408**. If the streamlets are not stored **608** locally, the streamlet request is passed to the network controller module **406**. The network controller module **406** then requests **616** the streamlet from the web server **116**. Once the streamlet is received, the network controller module **406** passes the streamlet to the streamlet cache module **404**. The streamlet cache module **404** archives **618** the streamlet. Alternatively, the streamlet cache module **404** then archives **618** the streamlet and passes the streamlet to the agent controller module **402**, and the method **600** then continues from operation **610** as described above.

Referring now to FIG. **7**, shown therein is a schematic flow chart diagram illustrating one embodiment of a method **700** for requesting streamlets within a adaptive-rate shifting content streaming environment in accordance with the present invention. The method **700** may be used in one embodiment as the operation **611** of FIG. **6**. The method **700** starts and the agent controller module **402** receives **704** a streamlet as described above with reference to FIG. **6**. The agent controller module **402** then monitors **706** the receive time of the requested streamlet. In one embodiment, the agent controller module **402** monitors the time intervals Δ between successive receive times for each streamlet response. Ordering of the responses in relation to the order of their corresponding requests is not relevant.

Because network behavioral characteristics fluctuate, sometimes quite suddenly, any given Δ may vary substantially from another. In order to compensate for this fluctuation, the agent controller module **402** calculates **708** a performance ratio r across a window of n samples for streamlets of playback length S. In one embodiment, the performance ratio r is calculated using the equation

$$r = S\frac{n}{\sum_{i=1}^{n} \Delta_i},$$

Due to multiple simultaneous streamlet processing, and in order to better judge the central tendency of the performance ratio r, the agent control module **402** may calculate a geometric mean, or alternatively an equivalent averaging algorithm, across a window of size m, and obtain a performance factor ϕ:

$$\varphi_{current} = \left(\prod_{j=1}^{m} r_j\right)^{\frac{1}{m}}.$$

The policy determination about whether or not to upshift **710** playback quality begins by comparing $\phi_{current}$ with a trigger threshold Θ_{up}. If $\phi_{current} \geq \Theta_{up}$, then an up shift to the next, higher quality stream may be considered **716**. In one embodiment, the trigger threshold Θ_{up} is determined by a combination of factors relating to the current read ahead margin (i.e. the amount of contiguously available streamlets that have been sequentially arranged by the staging module **409** for presentation at the current playback time index), and a minimum safety margin. In one embodiment, the minimum safety margin may be 24 seconds. The smaller the read ahead margin, the larger Θ_{up} is to discourage upshifting until a larger read ahead margin may be established to withstand network disruptions. If the agent controller module **402** is able to sustain **716** upshift quality, then the agent controller module **402** will upshift **717** the quality and subsequently request higher qualify streams. The determination of whether use of the higher quality stream is sustainable **716** is made by comparing an estimate of the higher quality stream's performance factor, ϕ_{higher}, with Θ_{up}. If $\phi_{higher} \geq \Theta_{up}$ then use of the higher quality stream is considered sustainable. If the decision of whether or not the higher stream rate is sustainable **716** is "no," the agent control module **402** will not attempt to upshift **717** stream quality. If the end of the stream has been reached **714**, the method **618** ends **716**.

If the decision on whether or not to attempt upshift **710** is "no", a decision about whether or not to downshift **712** is made. In one embodiment, a trigger threshold Θ_{down} is

defined in a manner analogous to Θ_{up}. If $\phi_{current} > \Theta_{down}$ then the stream quality may be adequate, and the agent controller module **402** does not downshift **718** stream quality. However, if $\phi_{current} \leq \Theta_{down}$, the agent controller module **402** does downshift **718** the stream quality. If the end of the stream has not been reached **714**, the agent controller module **402** begins to request and receive **704** lower quality streamlets and the method **618** starts again. Of course, the above described equations and algorithms are illustrative only, and may be replaced by alternative streamlet monitoring solutions.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. An end user station for adaptive-rate content streaming of digital content from a video server over a network, the end user station comprising:

a media player operating on the end user station configured to stream a video from the video server via at least one transmission control protocol (TCP) connection over the network, wherein multiple different copies of the video encoded at different bit rates are stored on the video server as multiple sets of files, wherein each of the files yields a different portion of the video on playback, wherein the files across the different copies yield the same portions of the video on playback, and wherein each of the files comprises a time index such that the files whose playback is the same portion of the video for each of the different copies have the same time index in relation to the beginning of the video, and wherein the media player streams the video by:

requesting a plurality of sequential files of one of the copies from the video server based on the time indexes;

automatically requesting from the video server subsequent portions of the video by requesting for each such portion one of the files from one of the copies dependent upon successive determinations by the media player to shift the playback quality to a higher or lower quality one of the different copies, the automatically requesting including repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the set of one or more factors relate to the performance of the network;

making the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies determined sustainable at that time so that the media player upshifts to a higher quality one of the different copies when the factor is greater than a first threshold and downshifts to a lower quality one of the different copies when the factor is less than a second threshold; and

presenting the video by playing back the requested media files with the media player on the end user station in order of ascending playback time.

2. The end user station of claim **1**, wherein the at least one TCP connection comprises multiple Transmission Control protocol (TCP) connections with the content server.

3. The end user station of claim **1**, wherein the media player is configured to generate the factor according to the responses to segment requests.

4. The end user station of claim **1**, wherein the media player is configured to upshift to the higher quality copy when the factor is greater than the first threshold and the media player determines the higher quality playback can be sustained according to a combination of factors.

5. The end user station of claim **1** wherein the media player is configured to upshift to the higher quality copy when the performance factor is greater than the first threshold and the media player determines that the higher quality playback can be sustained according to an amount of contiguously available files stored by the media player.

6. The end user station of claim **1**, wherein the media player is further configured to anticipate file requests and to pre-request files to enable fast-forward, skip randomly, and rewind functionality.

7. The end user station of claim **1**, wherein the media player is configured to initially request low quality files to enable instant playback of the video, and to subsequently upshift to a better quality copy of the video according to the performance factor.

8. A method executable by an end user station to present rate-adaptive streams received via at least one transmission control protocol (TCP) connection with a server over a network, the method comprising;

streaming, by a media player operating on the end user station, a video from the server via the at least one TCP connection over the network, wherein multiple different copies of the video encoded at different bit rates are stored as multiple sets of files on the server, wherein each of the files yields a different portion of the video on playback, wherein the files across the different copies yield the same portions of the video on playback, and wherein each of the files comprises a time index such that the files whose playback is the same portion of the video for each of the different copies have the same time index in relation to the beginning of the video, and wherein the streaming comprises:

requesting by the media player a plurality of sequential files of one of the copies from the server based on the time indexes;

automatically requesting by the media player from the server subsequent portions of the video by requesting for each such portion one of the files from one of the copies dependent upon successive determinations by the media player to shift the playback quality to a higher or lower quality one of the different copies, the automatically requesting including repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the factor relates to the performance of the network; and

making the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies determined sustainable at that time, wherein the making the successive determinations to shift comprises upshifting to a higher quality one of the different copies when the at least one factor is greater than a first threshold and downshifting to a lower quality one of the different copies when the at least one factor is less than a second threshold; and

presenting the video by playing back the requested media files with the media player on the end user station in order of ascending playback time.

9. The method of claim **8**, wherein the at least one TCP connection comprises a plurality of different TCP connections, and wherein the requesting the plurality of sequential

files includes requesting sub-parts of the files over different ones of the plurality of different TCP connections, and wherein said presenting includes reassembling the files from the received sub-parts.

10. The method of claim **8**, wherein said making the successive determinations to shift comprises:

determining if the higher quality playback can be sustained.

11. The method of claim **8**, wherein the at least one TCP connection comprises a plurality of different TCP connections, and wherein the automatically requesting includes requesting sub-parts of the files over different ones of the plurality of TCP connections, and wherein said presenting includes reassembling the files from the received sub-parts, and wherein the factor is indicative of the available bandwidth of the plurality of TCP connections.

12. The method of claim **8**, wherein the factor is indicative of latency of the requested files, wherein the latency is a time measure between when one of the requests is issued and a time that response data of the request begins to arrive at the end user station.

13. The method of claim **8**, wherein the factor is indicative of time intervals between successive receive times for each response to the requested files.

14. The method of claim **8**, wherein the factor is indicative of delays or losses in one or more of the at least one TCP connection.

15. The method of claim **8**, wherein the server is a web server, and wherein the files are requested from the web server using Hyper Text Transfer Protocol (HTTP) messages sent via the at least one TCP connection.

16. The method of claim **8**, wherein the server comprises a cache server of a network infrastructure operator.

* * * * *

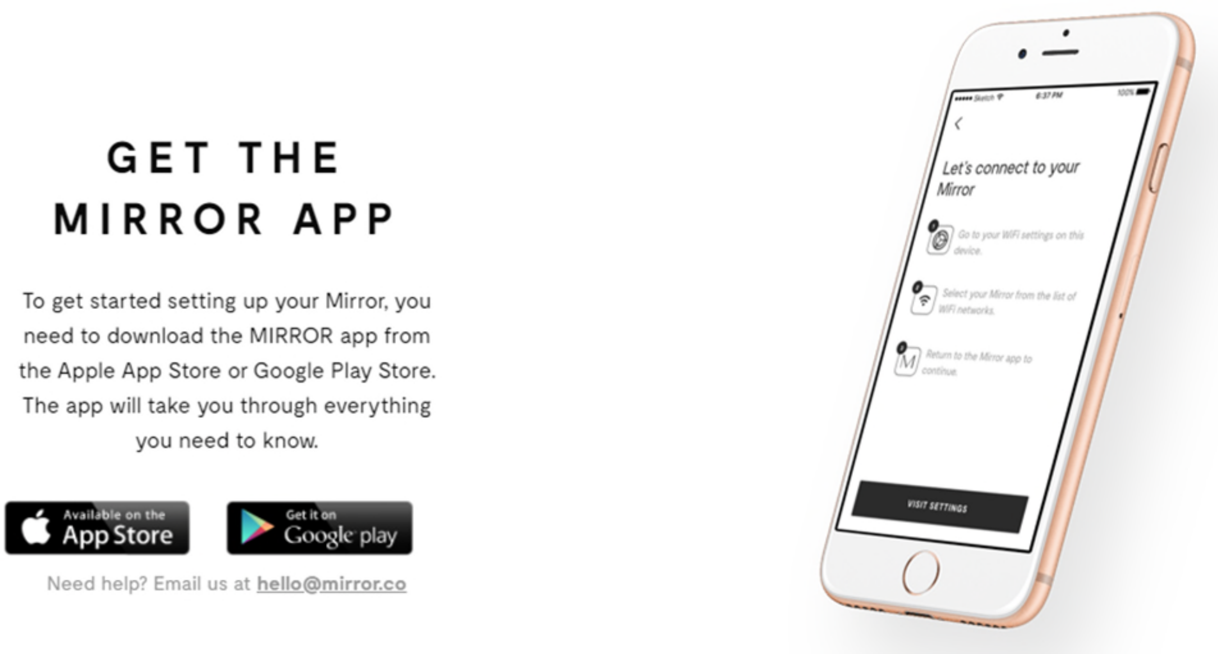
EXHIBIT A-1

U.S. Patent No. 9,407,564 to Mirror

The following claim chart shows exemplary aspects of the Mirror Application and Mirror Device that infringe the claims below. The chart is exemplary and should not be read to limit DISH's claims against Mirror to the specific products or services described below. The chart should also not be read to limit DISH's claims to the patent claims charted below. Nor should the chart below be read to limit how the Mirror Application and Mirror Devices infringe the claims below.

Claim	Claim Limitation	Example Infringement Evidence
1	1. An end user station for adaptive-rate content streaming of digital content from a video server over a network, the end user station comprising:	The Mirror Application is software that permits "an end user station" to perform "adaptive-rate content streaming of digital" live and on-demand "content from a video server over a network." The Mirror Application is executable by devices that that are end user stations and it obtains streams of selected digital content for adaptive-rate streaming. The streams are obtained by the Mirror Application over a network. The images in this chart of the Mirror Application are from the Mirror Application running on an Apple iPhone XS (Mirror's iOS Application) and connected to the Internet via TCP/IP protocols. In addition, the Mirror Application is available to run on other devices. Unless otherwise noted, each of those devices is an "end user station for adaptive-rate content streaming of digital content from a video server over a network." **MIRROR DIGITAL OVERVIEW** Access all of MIRROR's Live and On Demand Classes 24/7 from your phone, tablet, and smart TV devices. This feature is included for free with the MIRROR Membership and access is available immediately (whether you already have your Mirror or are awaiting delivery). Please note, it is not currently possible to purchase MIRROR Digital if you do not own a Mirror. https://mirror.kustomer.help/en_us/mirror-digital-overview-B1pnVd8UU

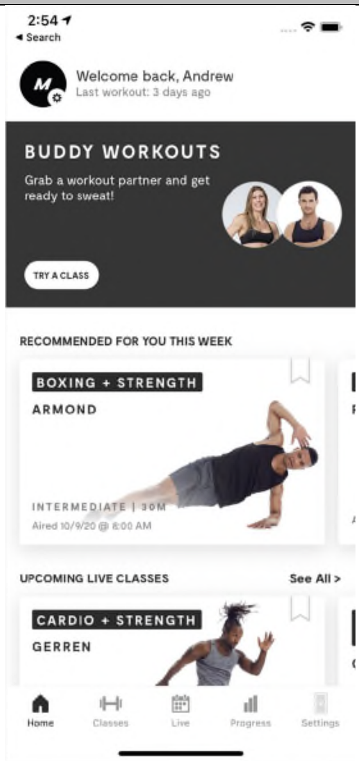
USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 https://www.mirror.co/app.

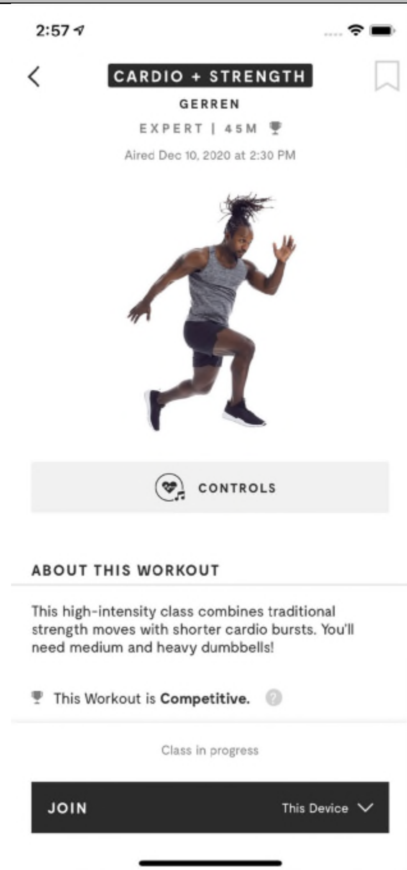
USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		MIRROR APP The MIRROR App allows you to access and customize the Mirror experience. The MIRROR App is available for both iOS and Android! • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-app--S1dDC_tYm.

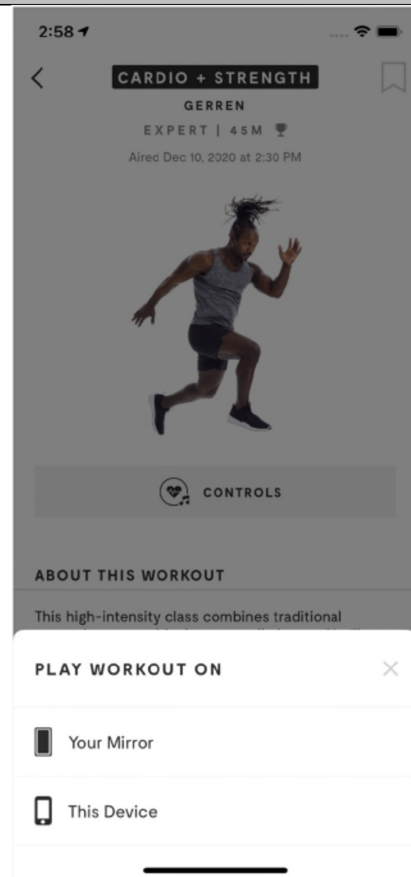
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Apple App Store When launched, the Mirror Application displays a main menu:

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application The main menu of the Mirror Application displays on-demand and live classes that are each "digital content." The "Live" section of the Mirror Application main menu displays a preview of ongoing and upcoming live digital content. The "Classes" section of the Mirror Application main menu displays a preview of on-demand digital content. Selecting a class causes the digital content to stream from the Mirror Serve(s) over the Internet and playback on the Mirror Application. Selecting a class causes the Mirror Application to provide options to stream the class to a variety of end user stations, including the iOS device that the Mirror Application is executing on or the separate Mirror Device.

USP 9,407,564 to Mirror

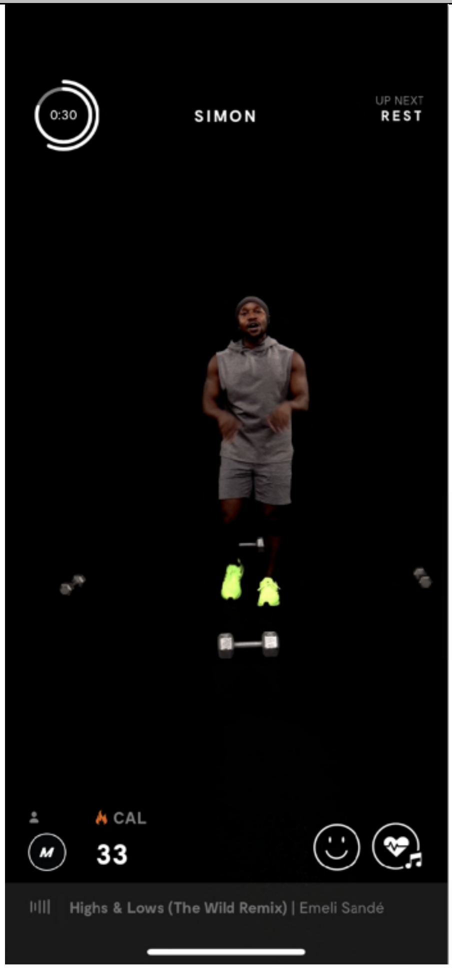
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application

USP 9,407,564 to Mirror

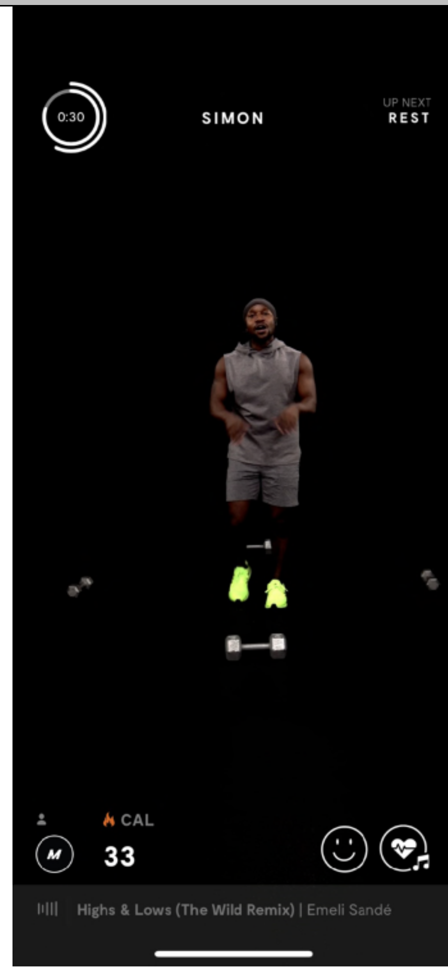
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting "Your Mirror" causes the digital content to be streamed on the user's Mirror device, which is connected to the Internet via TCP/IP protocols.

USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Alternatively, selecting "This Device" causes the digital content to be streamed on the user's iOS device:

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application As set forth above, Mirror Devices are "an end user station for adaptive-rate content streaming of digital content over a network from a video server over a network." The Mirror Devices obtain streams of

Claim	Claim Limitation	Example Infringement Evidence
		selected digital content for adaptive-rate content streaming. The streams are obtained over a network, specifically the Internet using TCP/IP protocols. As described in greater detail below, the Mirror Application and Mirror Devices operate in the same or a similar way to enable "adaptive-rate content streaming of digital content from a video server over a network."
	a media player operating on the end user station configured to stream a video from the video server via at least one transmission control protocol (TCP) connection over the network,	The Mirror Application includes a "media player" that "operat[es] on the end user station" executing the Mirror Application. For example, when digital content such as a class is selected, the Mirror Application launches a media player that "configured to stream a video from the video server via at least one transmission control protocol (TCP) connection over the network," as shown below.

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application The Mirror Devices include a "media player" that "operat[es] on the" Mirror Device "end user station." For example, when the "Your Mirror" option is selected in the Mirror Application after selecting a class,

Claim	Claim Limitation	Example Infringement Evidence
		the Mirror Device launches a media player that is "configured to stream a video from the video server(s) via at least one transmission control protocol (TCP) connection over the network," as shown below. 

Claim	Claim Limitation	Example Infringement Evidence
		As shown in greater detail below, the digital content is streamed to the Mirror Application and Mirror Devices.

	wherein multiple different copies of the video encoded at different bit rates are stored on the video server as multiple sets of files,	There are "multiple different copies of the video encoded at different bitrates … stored on the video server as multiple sets of files."
		
		For the following test, a live video was selected. In the test, an iPhone 11 running the Mirror Application makes an HTTPS GET request for a master playlist named "**playlist.m3u8**" that specifies the "multiple different copies of the video encoded at different bitrates" and provides links to the playlists for the multiple different copies of the video that are available. In response to the request, the following master playlist file named "playlist.m3u8" is returned.
		
		`#EXTM3U`
		`#EXT-X-VERSION:3`
		`#EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080`
		`../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720`
		`../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480`
		`../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360`
		`../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288`
		`../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180`
		`../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8`
		
		Filename: playlist.m3u8.

Claim	Claim Limitation	Example Infringement Evidence
		This is a master playlist file according to the HLS specification.[1] The playlist shows six different copies of the video, denoted by each #EXT-X-STREAM-INF tag at the following bandwidths: • 6434112 (referred to herein as "**6434112 Bandwidth**") • 864048 (referred to herein as "**864048 Bandwidth**") • 403824 (referred to herein as "**403824 Bandwidth**") • 367728 (referred to herein as "**367728 Bandwidth**") • 312832 (referred to herein as "**312832 Bandwidth**") • 249664 (referred to herein as "**249664 Bandwidth**") For each of the copies, the master playlist provides a link to a playlist file for the specified copy of the selected video at a particular bandwidth and resolution, which is called a "variant" in HLS. Each of the playlist files for each of the different copies of the video further include links to segments or streamlets of the video for the respective bandwidth and resolution of the copy. For example, the Mirror Application issued a request for the variant playlist file corresponding to the **6434112 Bandwidth** copy of the video, which is named "**chunklist.m3u8**." That file, including the links to the streamlets associated with that copy, is shown below.

[1] RFC 8216 (HLS Live Streaming), Section 4.3.4 (Master Playlist Tags)

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1232
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:08.356Z
#EXTINF:2.0,
r4vhrugx/00000000/media_1232.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1233.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1234.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1235.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1236.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1237.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1238.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1239.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1240.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1241.ts
#EXTINF:2.0,
```

Filename: https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8

The variant playlist file corresponding to the **403824 Bandwidth** copy of the video, titled "**chunklist.m3u8**," is shown below.</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1238 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z #EXTINF:2.0, fbd862nq/00000000/media_1238.ts #EXTINF:2.0, fbd862nq/00000000/media_1239.ts #EXTINF:2.0, fbd862nq/00000000/media_1240.ts #EXTINF:2.0, fbd862nq/00000000/media_1241.ts #EXTINF:2.0, fbd862nq/00000000/media_1242.ts #EXTINF:2.0, fbd862nq/00000000/media_1243.ts #EXTINF:2.0, fbd862nq/00000000/media_1244.ts #EXTINF:2.0, fbd862nq/00000000/media_1245.ts #EXTINF:2.0, fbd862nq/00000000/media_1246.ts #EXTINF:2.0, fbd862nq/00000000/media_1247.ts #EXTINF:2.0, fbd862nq/00000000/media_1248.ts Filename: **chunklist.m3u8** Each of the segments or streamlets corresponding to each of the different copies of the video are also stored, as shown by the Mirror Application issuing a GET request for the "media_1238.ts" streamlet.

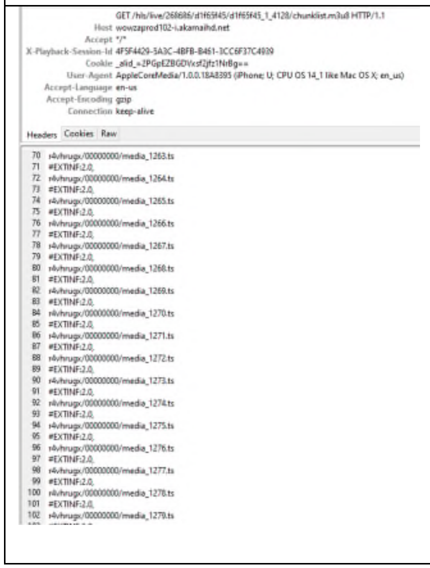
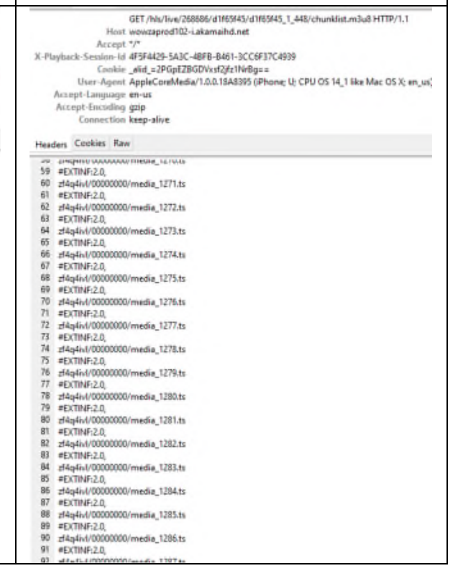
<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>The various versions of the segments with different bandwidths and different resolutions are stored and then accessed based on requests from the Mirror Application and Mirror Devices. The 6434112 Bandwidth version of the segments of the program are encoded and stored in a directory (e.g., r4vhrugx/00000000/”) and the 403824 Bandwidth version of the segments of the program are encoded and stored in a directory (e.g., “fbd862nq/00000000/”). As explained above, these versions have different bit rates which are identified in the master playlist file (i.e., filename: playlist.m3u8). The identical segments of the filenames in each directory demonstrate that these segments are copies of the same video.</td></tr>
<tr><td></td><td>wherein each of the files yields a different portion of the video on playback,</td><td>Each of the .ts files (e.g., segments or streamlets) “yields a different portion of the video on playback.” An excerpt of the Charles Proxy sequence listing is provided below and shows the Mirror Application requesting and receiving different, sequential 2 second segments of the program to playback different portions of the video. The sequence listing below shows the Mirror Application requesting and receiving three sequential segments of the program “media_1274.ts,” “media_1275.ts,” and “media_1276.ts.” The separate files are separate segments for different time indexes and portions of the video on playback. As discussed above, there are multiple versions of each of these files for each time index.

<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1274.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1275.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1276.ts</td></tr>
</table>

On information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.</td></tr>
<tr><td></td><td>wherein the files across the different copies yield the same portions of the</td><td>The “files across the different copies yield the same portions of the video on playback” on the Mirror Application and Mirror Devices. As described above, each of the playlists includes links to the files with the same video content at different bandwidths and resolutions.</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
	video on playback, and	For example, each variant playlist includes multiple streamlets, including a streamlet with the filename ending in "**media_1274.ts**" A comparison of the **6434112 Bandwidth**, **403824 Bandwidth**, and **249664 Bandwidth** copies from above shows that each of the playlists includes the "**media_1274.ts**" segment. On information and belief, playlists for the other copies also include this segment. As discussed above, each streamlet corresponds to a portion of the video on playback. Notably, each bitrate copy of the **media_1274.ts** segment has a duration of 2 seconds (as noted in each line beginning with #EXTINF and corresponds to the same time index, thereby yielding "the same portions of the video on playback." Upon information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application.
	wherein each of the files comprises a time index such that the files whose playback is the same portion of the video for each of the different copies have the same time index in relation to the beginning of the video, and wherein the media player streams the video by:	As shown above, "each of the files comprises a time index such that the files whose playback is the same portion of the video for each of the different copies have the same time index in relation to the beginning of the video." As described above, and as explained further below, each of the playlists includes links to files with the same video content at different bandwidths and resolutions. For example, compare the segment files in the **6434112 Bandwidth**, **403823 Bandwidth**, and **249664 Bandwidth** copies of the video:

USP 9,407,564 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>6434112 Bandwidth</th><th>403824 Bandwidth</th><th>249664 Bandwidth</th></tr>
<tr><td></td><td></td><td></td></tr>
</table>

These three playlists refer to segments with identical filename conventions (“media_#.ts”) and identical lengths (in seconds) with different file path prefixes. In particular, for example, file names containing “media_1275.ts” are all 2.0 seconds in length and are available for both bandwidths (and corresponding resolutions and bit rates).
</td></tr>
<tr><td></td><td>requesting a plurality of sequential files of one of the copies from the video server based on the time indexes;</td><td>The Mirror Application and Mirror Devices stream the video by “requesting a plurality of sequential files of one of the copies from the video server based on the time indexes.” The requests are sent from the Mirror Application and/or Mirror Devices.

The sequence listing below shows the Mirror Application requesting and receiving three sequential segments of the 249664 Bandwidth copy of the video: “media_1274.ts,” “media_1275.ts,” and “media_1276.ts.” The files are sequential and separate segments for different time indexes. As discussed above, there are multiple versions of each of these files for each time index.</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		<table><tr><th>Method</th><th>Host</th><th>Path</th></tr><tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1274.ts</td></tr><tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1275.ts</td></tr><tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1276.ts</td></tr></table> On information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application, as shown above.
	automatically requesting from the video server subsequent portions of the video by requesting for each such portion one of the files from one of the copies dependent upon successive determinations by the media player to shift the playback quality to a higher or lower quality one of the different copies,	The Mirror Application and Mirror Devices stream the video by "automatically requesting from the video server subsequent portions of the video by requesting for each such portion one of the files from one of the copies dependent upon successive determinations by the media player to shift the playback quality to a higher or lower quality one of the different copies." In order to demonstrate the bandwidth adaptation ("shift[ing] the playback quality"), the throttling feature of the Charles Proxy application was used to limit the Mirror Application's bandwidth to approximate a slower speed, and then the throttling setting was removed. When the bandwidth for the Mirror Application is reduced, the Mirror Application engages in adaptation to shift playback quality to a lower quality one of the different copies by requesting a lower bit rate version of the content at a subsequent time index, and when bandwidth for the Mirror Application is unconstrained, the Mirror Application engages in adaptation to shift playback quality to a higher quality one of the different copies in a similar way. This behavior demonstrates the automatic requesting dependent upon successive determinations by the media player to shift the playback quality. Recall from the discussion about the playlist.m3u8 file that the variant playlists and segments are stored in different directories, based upon the resolution and bandwidth. <table><tr><th>Bandwidth</th><th>Playlist Filename</th></tr><tr><td>6434112</td><td>d1f65f45_1_4128/chunklist.m3u8</td></tr><tr><td>864048</td><td>d1f65f45_1_2728/chunklist.m3u8</td></tr></table>

Claim	Claim Limitation	Example Infringement Evidence
		403824 — d1f65f45_1_1728/chunklist.m3u8 367728 — d1f65f45_1_1152/chunklist.m3u8 312832 — d1f65f45_1_640/chunklist.m3u8 249664 — d1f65f45_1_448/chunklist.m3u8 The chosen resolutions of the Mirror Application can be determined based on which playlist and associated segments of the video are retrieved. A portion of the Charles Proxy sequence listing shows the requested and retrieved files while bandwidth was constrained and after the bandwidth was unconstrained is shown below.

403824	d1f65f45_1_1728/chunklist.m3u8
367728	d1f65f45_1_1152/chunklist.m3u8
312832	d1f65f45_1_640/chunklist.m3u8
249664	d1f65f45_1_448/chunklist.m3u8

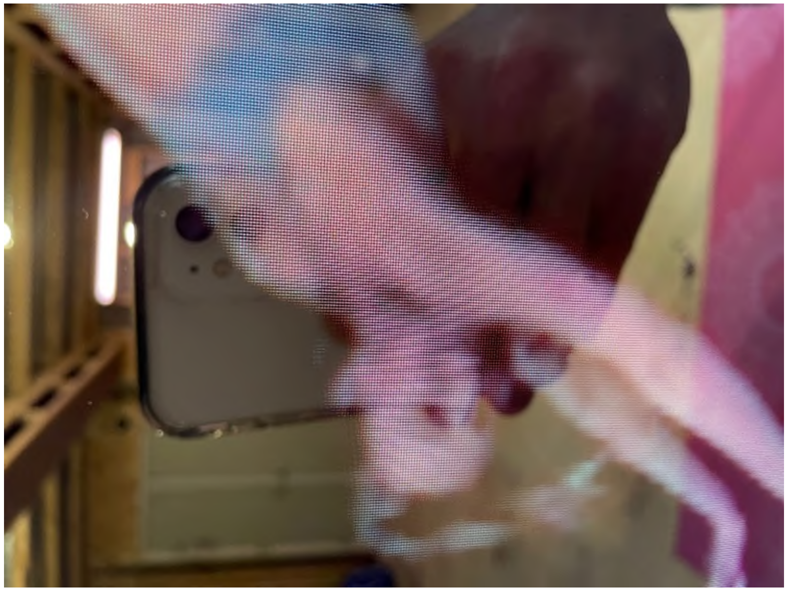
Method	Host	Path
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1277.ts
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_640/3fyynadk/00000000/media_1278.ts

When the Mirror Application operates at an unconstrained bandwidth, the Mirror Application transitions from a lower bitrate version of the video (e.g., the **249664 Bandwidth** version) to a higher bitrate version of the video (e.g., the **312832 Bandwidth** version). Between these two segments, the Mirror Application requests and receives a variant playlist file for the **312832 Bandwidth** version of the video ("/hls/live/268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8").

As demonstrated above, the Mirror Application requests and receives the lower resolution encoded files when the bandwidth is constrained and requests and receives the higher resolution encoded files when the bandwidth is unconstrained. In this way, the Mirror Application adapts subsequent segment requests based on successive determinations to shift the playback to a higher or lower quality encoding.

Claim	Claim Limitation	Example Infringement Evidence
		On information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above. For example, during a test of the Mirror Devices, the Mirror Devices successively determined to and shifted from a higher quality copy to a lower quality copy of the video when the bandwidth was constrained and then shifted back to a higher quality copy from the lower quality copy when the bandwidth was unconstrained, as shown below by the test capturing the shifting through multiple different copies of the video at varying qualities. First version: 

Claim	Claim Limitation	Example Infringement Evidence
		Second version: 

Claim	Claim Limitation	Example Infringement Evidence
		Third version: 
	the automatically requesting including repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies,	The automatically requesting by the Mirror Application and Mirror Devices includes "repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the set of one or more factors relate to the performance of the network." The repeated generation of a factor indicative of the current ability to sustain streaming of the video using the files from different ones of the copies is demonstrated by testing where bandwidth available to the Mirror Application and Mirror Devices is throttled and unthrottled and the playback quality automatically shifted in accordance with the player's ability to sustain the stream under the bandwidth constraints, as was shown above. In addition, the Mirror Devices run the Android operating system. ExoPlayer is a ubiquitous HLS video player for Android. ExoPlayer2 provide a DefaultBandwidthMeter class that "[e]stimates bandwidth by

Claim	Claim Limitation	Example Infringement Evidence
	wherein the set of one or more factors relate to the performance of the network;	listening to data transfers." https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/upstream/DefaultBandwidthMeter.html The ExoPlayer documentation further explains that "[t]he bandwidth estimate is calculated using a SlidingPercentile and is updated each time a transfer ends. The initial estimate is based on the current operator's network country code or the locale of the user, as well as the network connection type. This can be configured in the DefaultBandwidthMeter.Builder." *Id*. ExoPlayer's DefaultBandwidthMeter therefore repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the set of one or more factors relate to the performance of the network. The estimated bandwidth from the DefaultBandwidthMeter is then used by the AdaptiveTrackSelection.Factory class to determine whether to change to a higher or lower version of the stream. https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/trackselection/AdaptiveTrackSelection.Factory.html The AdaptiveTrackSelection.Factory class uses the following parameters to determine whether to switch: minDurationForQualityIncreaseMs - The minimum duration of buffered data required for the selected track to switch to one of higher quality. maxDurationForQualityDecreaseMs - The maximum duration of buffered data required for the selected track to switch to one of lower quality. minDurationToRetainAfterDiscardMs - When switching to a track of significantly higher quality, the selection may indicate that media already buffered at the lower quality can be discarded to speed up the switch. This is the minimum duration of media that must be retained at the lower quality.

Claim	Claim Limitation	Example Infringement Evidence
		bandwidthFraction - The fraction of the available bandwidth that the selection should consider available for use. Setting to a value less than 1 is recommended to account for inaccuracies in the bandwidth estimator. *Id*.
	making the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies determined sustainable at that time so that the media player upshifts to a higher quality one of the different copies when the factor is greater than a first threshold and downshifts to a lower quality	The Mirror Application and Mirror Devices stream the video by "making the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies determined sustainable at that time." This is done "so that the media player upshifts to a higher quality one of the different copies when the factor is greater than a first threshold and downshifts to a lower quality one of the copies when the factor is less than a second threshold." The shifts between the higher and lower quality versions of the selected video demonstrates the determination to shift playback quality "based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies determined sustainable at that time. As noted above, the Mirror Application and Mirror Devices shift between playback quality based on a factor that includes, for example, bandwidth limitations to enable continuous playback of the video. Further, the Mirror Application and Mirror Devices request a lower quality content when bandwidth is constrained, as shown herein. Accordingly, the Mirror Application and Mirror Devices continue to request the highest quality content that is sustainable when doing so. Making the successive determinations to shift comprises upshifting to a higher quality one of the different copies when the at least one factor "is greater than a first threshold" and downshifting to a lower quality one of the different copies when the at least one factor "is less than second threshold." This is demonstrated at least by the Mirror Application shifting to the highest-quality version it is able to be based on bandwidth constraints. When the Mirror Application operates at unconstrained bandwidth, the Mirror Application transitions from a lower-bitrate version of the video (**249664 Bandwidth** version) to a higher-bitrate version of the video (**367728 Bandwidth** version). Between these two successive segments, the Mirror Application requests and receives a **367728 Bandwidth** variant playlist file.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td>one of the different copies when the factor is less than a second threshold; and</td><td>

Method	Host	Path
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
..		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1279.ts
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1152/bmit701z/00000000/media_1278.ts

As demonstrated above, the Mirror Application requests and receives the higher resolution segment when operating at unconstrained bandwidth. Accordingly, the Mirror Application determined that the performance ratio exceeded a first threshold value prior to requesting the higher quality segments.

A portion of the Charles Proxy sequence listing below shows the requested and retrieved files prior to and just after bandwidth was constrained is shown below.

Method	Host	Path
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8
..		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts

When the Mirror Application operates at a constrained bandwidth, the Mirror Application transitions from a higher-bitrate version of the video (the **403824 Bandwidth** version) to a lower-bitrate version of the
</td></tr>
</table>

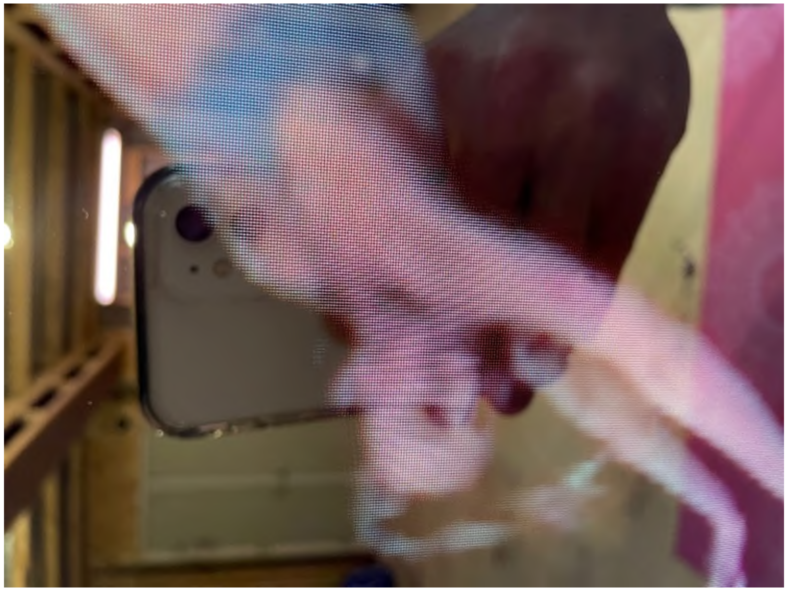
Claim	Claim Limitation	Example Infringement Evidence
		video (the **249664 Bandwidth** version). Between the two segments, the Mirror Application requests and receives a **249664 Bandwidth** variant playlist file of the video. As demonstrated above, the Mirror Application requests and receives the lower resolution encoded files while the bandwidth is constrained. Accordingly, the Mirror Application determines the performance ratio was less than a second threshold value prior to requesting lower quality segments. In operation, when full bandwidth is returned, the Mirror Application again requests and plays a higher bitrate segment of the video, which is in response the factor being greater than a threshold. Upon information and belief, the Mirror Devices operates in the same or substantially the same way. For example, when bandwidth is unconstrained, the Mirror Devices shift to the highest quality copy of the video that is sustainable.

USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		First version:  For the current test, when the bandwidth was limited, the Mirror Device subsequently shifted to a lower quality copy of the video.

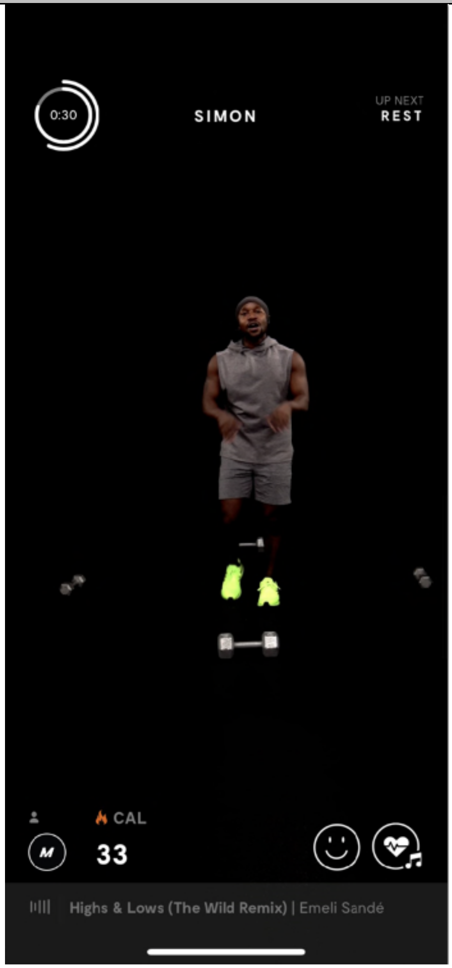
USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Second version: 

Claim	Claim Limitation	Example Infringement Evidence
		Third version:  When the bandwidth constraint was removed, the Mirror Device subsequently returned to a higher-bandwidth version shown above. The Mirror Devices run the Android operating system. ExoPlayer is a ubiquitous HLS video player for Android. ExoPlayer2 provide a DefaultBandwidthMeter class that "[e]stimates bandwidth by listening to data transfers." https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/upstream/DefaultBandwidthMeter.html. The ExoPlayer documentation further explains that "[t]he bandwidth estimate is calculated using a SlidingPercentile and is updated each time a transfer ends. The initial estimate is based on the current

Claim	Claim Limitation	Example Infringement Evidence
		operator's network country code or the locale of the user, as well as the network connection type. This can be configured in the DefaultBandwidthMeter.Builder." *Id*. ExoPlayer's DefaultBandwidthMeter therefore repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the set of one or more factors relate to the performance of the network. The estimated bandwidth from the DefaultBandwidthMeter is then used by the AdaptiveTrackSelection.Factory class to determine whether to change to a higher or lower version of the stream. https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/trackselection/AdaptiveTrackSelection.Factory.html The AdaptiveTrackSelection.Factory class uses the following parameters to determine whether to switch: minDurationForQualityIncreaseMs - The minimum duration of buffered data required for the selected track to switch to one of higher quality. maxDurationForQualityDecreaseMs - The maximum duration of buffered data required for the selected track to switch to one of lower quality. minDurationToRetainAfterDiscardMs - When switching to a track of significantly higher quality, the selection may indicate that media already buffered at the lower quality can be discarded to speed up the switch. This is the minimum duration of media that must be retained at the lower quality. bandwidthFraction - The fraction of the available bandwidth that the selection should consider available for use. Setting to a value less than 1 is recommended to account for inaccuracies in the bandwidth estimator. *Id*. ExoPlayer therefore makes the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies

Claim	Claim Limitation	Example Infringement Evidence
		determined sustainable at that time so that the media player upshifts to a higher quality one of the different copies when the factor is greater than a first threshold and downshifts to a lower quality one of the different copies when the factor is less than a second threshold.
	presenting the video by playing back the requested media files with the media player on the end user station in order of ascending playback time.	As shown above, the Mirror Application and Mirror Devices receive the playlist file that lists the .ts file segments in order of ascending playback time and the Mirror Application requests those same .ts files in order of ascending playback time. The Mirror Application and Mirror Devices then "present[] the video by playing back the requested media files with the media player on the end user station in order of ascending playback time." The Mirror Application and Mirror Devices playback the requested .ts files in order of ascending playback time after they are retrieved. For example, the media player of the Mirror Application includes a timer or timeline indicating that the media files are played back in order of ascending playback time in accordance with the timeline.

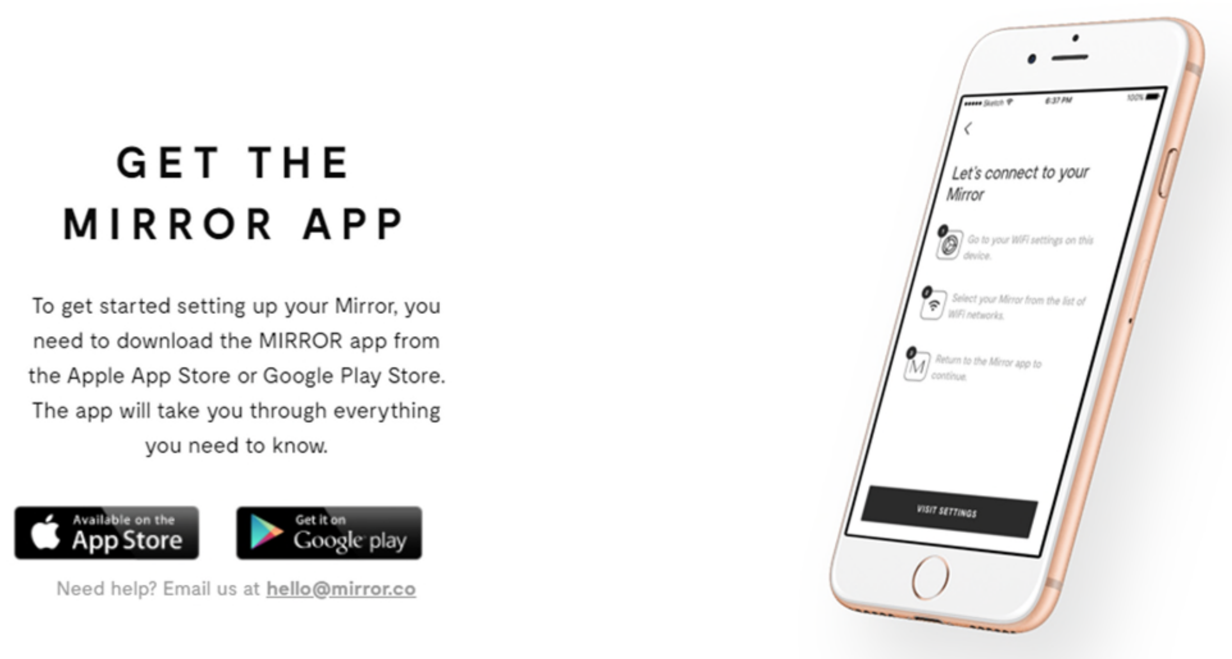
Claim	Claim Limitation	Example Infringement Evidence
		 For example, the media player of the Mirror Devices includes a timer or timeline indicating that the media files are played back in order of ascending playback time in accordance with the timeline.

USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		
8	8. A method executable by an end user station to present rate-adaptive streams received via at least one transmission control protocol (TCP) connection with a server over	The Mirror Application is software that is "executable by an end user station" and it executes a method "to present rate-adaptive streams received via at least one transmission control protocol (TCP) connection with a server over a network." The rate-adaptive streams are obtained over a network via at least one TCP connection. The images in this chart of the Mirror Application are from the Mirror Application running on an Apple iPhone XS (Mirror's iOS Application) connected to the Internet via TCP/IP protocols. In addition, the Mirror Application is available to run on other devices. Each of these devices is an "end user station."

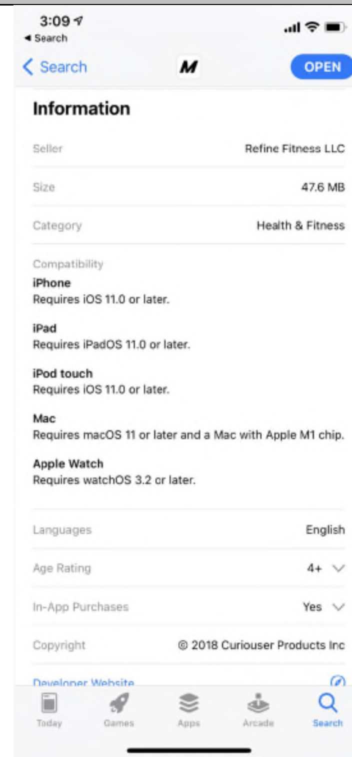
Claim	Claim Limitation	Example Infringement Evidence
	a network, the method comprising;	**MIRROR DIGITAL OVERVIEW** Access all of MIRROR's Live and On Demand Classes 24/7 from your phone, tablet, and smart TV devices. This feature is included for free with the MIRROR Membership and access is available immediately (whether you already have your Mirror or are awaiting delivery). Please note, it is not currently possible to purchase MIRROR Digital if you do not own a Mirror. https://mirror.kustomer.help/en_us/mirror-digital-overview-B1pnVd8UU **STREAMING MIRROR DIGITAL TO TV** You can stream classes from the MIRROR App to your television using below devices. The casting devices listed below have been tested by the MIRROR team and are compatible with MIRROR Digital. While additional devices may be compatible, MIRROR cannot guarantee functionality until the device has been thoroughly tested. We will continue to add to this list as MIRROR Digital develops! **iOS Casting Compatible Devices:** • Apple TV (all models except the first generation) • Airport Express • Chromecast • There are third-party apps you can use to stream to other devices, but they may not be supported in the MIRROR App **Android Casting Compatible Devices:** • Chromecast • Android TV • Google TV • Samsung TV https://mirror.kustomer.help/en_us/streaming-mirror-digital-to-tv-rJNcuu8I8

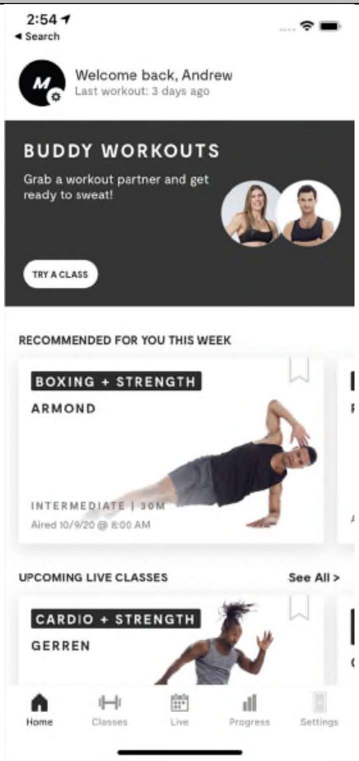
USP 9,407,564 to Mirror

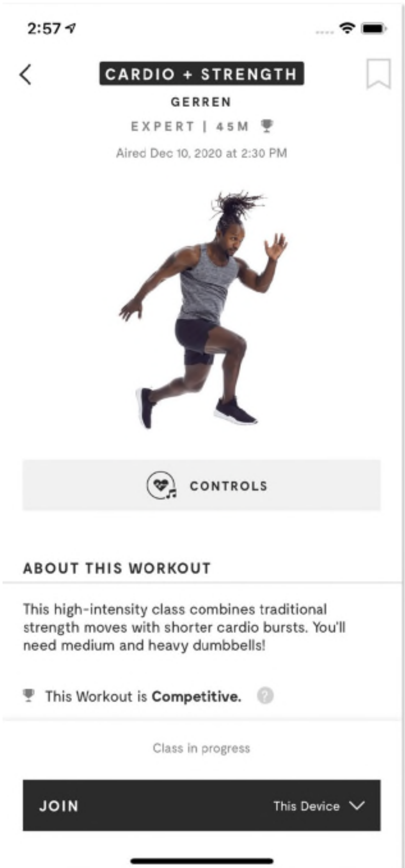
Claim	Claim Limitation	Example Infringement Evidence
		 https://www.mirror.co/app.

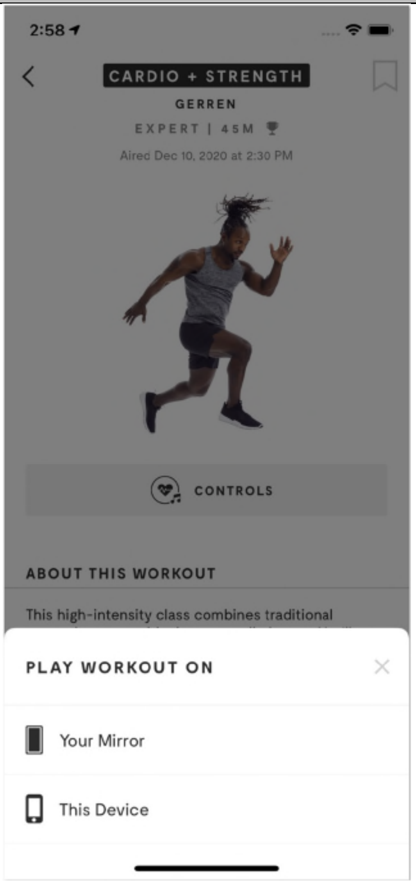
Claim	Claim Limitation	Example Infringement Evidence
		MIRROR APP The MIRROR App allows you to access and customize the Mirror experience. The MIRROR App is available for both iOS and Android! • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-app--S1dDC_tYm.

USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Apple App Store When launched, the Mirror Application displays a main menu:

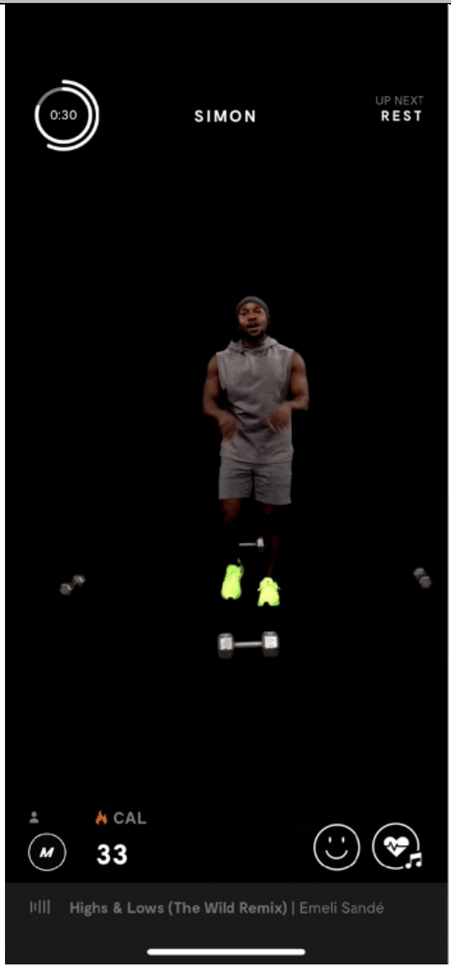
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application The main menu of the Mirror Application displays on-demand and live classes that represent "rate-adaptive streams." The "Live" section of the Mirror Application main menu displays a preview of ongoing and upcoming live adaptive-rate streams. The "Classes" section of the Mirror Application main menu displays a preview of on-demand rate-adaptive streams. Selecting a class causes the digital content to stream from the Mirror Serve(s) over the Internet and playback on the Mirror Application. Selecting a class causes the Mirror Application to provide options to stream the class to a variety of end user stations, including the iOS device that the Mirror Application is executing on or the separate Mirror Device.

Claim	Claim Limitation	Example Infringement Evidence
		Source: Mirror iOS Application  Source: Mirror iOS Application

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting "Your Mirror" causes the rate-adaptive stream to be presented on the user's Mirror device, which is connected to the Internet via TCP/IP protocols.

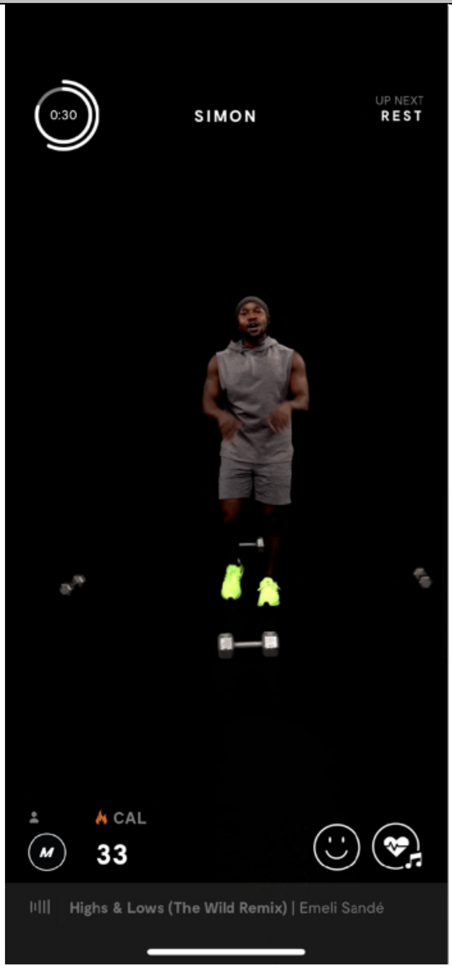
Claim	Claim Limitation	Example Infringement Evidence
		 Alternatively, selecting “This Device” causes the rate-adaptive stream to be presented on the user’s iOS device:

USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application As set forth above, Mirror Devices are "end user station[s]" that include software that is "executable" by the Mirror Device "to present rate-adaptive streams received via at least one transmission control

Claim	Claim Limitation	Example Infringement Evidence
		protocol (TCP) connection with a server over the network." The rate-adaptive streams are obtained over a network via at least one TCP connection. As described in greater detail below, the Mirror Application and Mirror Devices operate in the same or a similar way to execute a method to present-rate adaptive streams received via the at least one TCP connection over the Internet.
	streaming, by a media player operating on the end user station, a video from the server via the at least one TCP connection over the network,	The Mirror Application includes a "media player" that is "operating on the end user station" that executes the Mirror Application. For example, when a class is selected, the Mirror Application launches a media player and "stream[s] … a video from the server via at least one TCP connection over the network," as shown below.

USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application In particular, as will be shown in detail below, the Mirror Application sends HTTP GET requests transmitted by the TCP protocol for the purpose of requesting and presenting rate-adaptive streams.

Claim	Claim Limitation	Example Infringement Evidence
		The Mirror Devices include a "media player." When a class is selected, and "Your Mirror" is selected, a Mirror Device launch the media player and "stream[s] … a video from the server via the at least one TCP connection over the network," as shown below. 

Claim	Claim Limitation	Example Infringement Evidence
		As shown in greater detail below, the adaptive-rate stream is streamed to the Mirror Application and Mirror Devices.
	wherein multiple different copies of the video encoded at different bit rates are stored as multiple sets of files on the server,	There are "multiple different copies of the video encoded at different bitrates … stored on the video server as multiple sets of files." For the following test, a live video was selected. In the test, an iPhone 11 running the Mirror Application makes an HTTPS GET request for a master playlist named "**playlist.m3u8**" that specifies the "multiple different copies of the video encoded at different bitrates" and provides links to the playlists for the multiple different copies of the video that are available. In response to the request, the following master playlist file named "playlist.m3u8" is returned. #EXTM3U #EXT-X-VERSION:3 #EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080 ../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720 ../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480 ../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360 ../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288 ../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180 ../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 Filename: playlist.m3u8.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>This is a master playlist file according to the HLS specification.[2] The playlist shows six different copies of the video, denoted by each #EXT-X-STREAM-INF tag at the following bandwidths:

6434112 (referred to herein as “6434112 Bandwidth”)
864048 (referred to herein as “864048 Bandwidth”)
403824 (referred to herein as “403824 Bandwidth”)
367728 (referred to herein as “367728 Bandwidth”)
312832 (referred to herein as “312832 Bandwidth”)
249664 (referred to herein as “249664 Bandwidth”)

For each of the copies, the master playlist provides a link to a playlist file for the specified copy of the selected video at a particular bandwidth and resolution, which is called a “variant” in HLS. Each of the playlist files for each of the different copies of the video further include links to segments or streamlets of the video for the respective bandwidth and resolution of the copy. For example, the Mirror Application issued a request for the variant playlist file corresponding to the 6434112 Bandwidth copy of the video, which is named “chunklist.m3u8.” That file, including the links to the streamlets associated with that copy, is shown below.</td></tr>
</table>

[2] RFC 8216 (HLS Live Streaming), Section 4.3.4 (Master Playlist Tags)

Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1232 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:08.356Z #EXTINF:2.0, r4vhrugx/00000000/media_1232.ts #EXTINF:2.0, r4vhrugx/00000000/media_1233.ts #EXTINF:2.0, r4vhrugx/00000000/media_1234.ts #EXTINF:2.0, r4vhrugx/00000000/media_1235.ts #EXTINF:2.0, r4vhrugx/00000000/media_1236.ts #EXTINF:2.0, r4vhrugx/00000000/media_1237.ts #EXTINF:2.0, r4vhrugx/00000000/media_1238.ts #EXTINF:2.0, r4vhrugx/00000000/media_1239.ts #EXTINF:2.0, r4vhrugx/00000000/media_1240.ts #EXTINF:2.0, r4vhrugx/00000000/media_1241.ts #EXTINF:2.0, Filename: https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 The variant playlist file corresponding to the **403824 Bandwidth** copy of the video, titled "**chunklist.m3u8**," is shown below.

USP 9,407,564 to Mirror

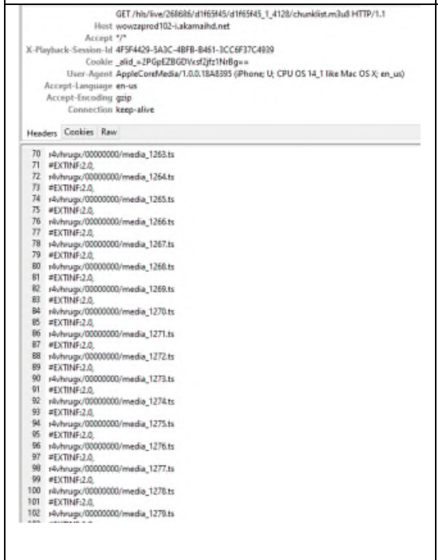
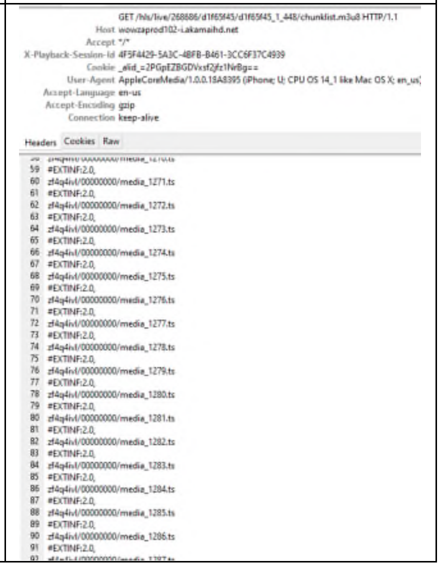
<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1238
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z
#EXTINF:2.0,
fbd862nq/00000000/media_1238.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1239.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1240.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1241.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1242.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1243.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1244.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1245.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1246.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1247.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1248.ts
Filename: chunklist.m3u8

Each of the segments or streamlets corresponding to each of the different copies of the video are also stored, as shown by the Mirror Application issuing a GET request for the “media_1238.ts” streamlet.</td></tr>
</table>

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>The various versions of the segments with different bandwidths and different resolutions are stored then accessed based on requests from the Mirror Application and Mirror Devices. The 6434112 Bandwidth version of the segments of the program are encoded and stored in a directory (e.g., “r4vhrugx/00000000/”), and the 403824 Bandwidth version of the segments of the program are encoded and stored in a directory (e.g., “fbd862nq/00000000/”). As explained above, these versions have different bit rates which are identified in the master playlist file (i.e., filename: playlist.m3u8). The identical segments of the filenames in each directory demonstrate that these segments are copies of the same video.</td></tr>
<tr><td></td><td>wherein each of the files yields a different portion of the video on playback,</td><td>Each of the .ts files (e.g., segments or streamlets) “yields a different portion of the video on playback.” An excerpt of the Charles Proxy sequence listing is provided below and shows the Mirror Application requesting and receiving different, sequential 2 second segments of the program to playback different portions of the video. The sequence listing below shows the Mirror Application requesting and receiving three sequential segments of the program “media_1274.ts,” “media_1275.ts,” and “media_1276.ts.” The separate files are separate segments for different time indexes and portions of the video on playback. As discussed above, there are multiple versions of each of these files for each time index.
<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1274.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1275.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1276.ts</td></tr>
</table>
On information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.</td></tr>
<tr><td></td><td>wherein the files across the different copies yield the same portions of the</td><td>The “files across the different copies yield the same portions of the video on playback” on the Mirror Application and Mirror Devices. As described above, each of the playlists includes links to the files with the same video content at different bandwidths and resolutions.

For example, each variant playlist includes multiple streamlets, including a streamlet with the filename ending in “media_1274.ts” A comparison of the 6434112 Bandwidth, 403824 Bandwidth, and 249664</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
	video on playback, and	**Bandwidth** copies from above shows that each of the playlists includes the "**media_1274.ts**" segment. On information and belief, playlists for the other copies also include this segment. As discussed above, each streamlet corresponds to a portion of the video on playback. Notably, each bitrate copy of the **media_1274.ts** segment has a duration of 2 seconds (as noted in each line beginning with #EXTINF and corresponds to the same time index, thereby yielding "the same portions of the video on playback." Upon information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application.
	wherein each of the files comprises a time index such that the files whose playback is the same portion of the video for each of the different copies have the same time index in relation to the beginning of the video, and wherein the streaming comprises:	As shown above, "each of the files comprises a time index such that the files whose playback is the same portion of the video for each of the different copies have the same time index in relation to the beginning of the video." As described above, and as explained further below, each of the playlists includes links to files with the same video content at different bandwidths and resolutions. For example, compare the segment files in the **6434112 Bandwidth**, **403823 Bandwidth**, and **249664 Bandwidth** copies of the video:

Claim	Claim Limitation	Example Infringement Evidence
		6434112 Bandwidth  **403824 Bandwidth**  **249664 Bandwidth**  These three playlists refer to segments with identical filename conventions ("media_#.ts") and identical lengths (in seconds) with different file path prefixes. In particular, for example, file names containing "media_1275.ts" are all 2.0 seconds in length and are available for both bandwidths (and corresponding resolutions and bit rates).
	requesting by the media player a plurality of sequential files of one of the copies from the server based on the time indexes;	The media player of the Mirror Application and Mirror Devices streams the video by "requesting … a plurality of sequential files of one of the copies from the server based on the time indexes." The requests are sent from the Mirror Application and/or Mirror Devices. The sequence listing below shows the Mirror Application requesting and receiving three sequential segments of the **249664 Bandwidth** copy of the video: "media_1274.ts," "media_1275.ts," and "media_1276.ts." The files are sequential and separate segments for different time indexes. As discussed above, there are multiple versions of each of these files for each time index.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1274.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1275.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1276.ts</td></tr>
</table>
On information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application, as shown above.</td></tr>
<tr><td></td><td>automatically requesting by the media player from the server subsequent portions of the video by requesting for each such portion one of the files from one of the copies dependent upon successive determinations by the media player to shift the playback quality to a higher or lower quality one of the different copies,</td><td>The media player of the Mirror Application and Mirror Devices streams the video by “automatically requesting … from the server subsequent portions of the video by requesting for each such portion one of the files from one of the copies dependent upon successive determinations by the media player to shift the playback quality to a higher or lower quality one of the different copies,” as shown below.

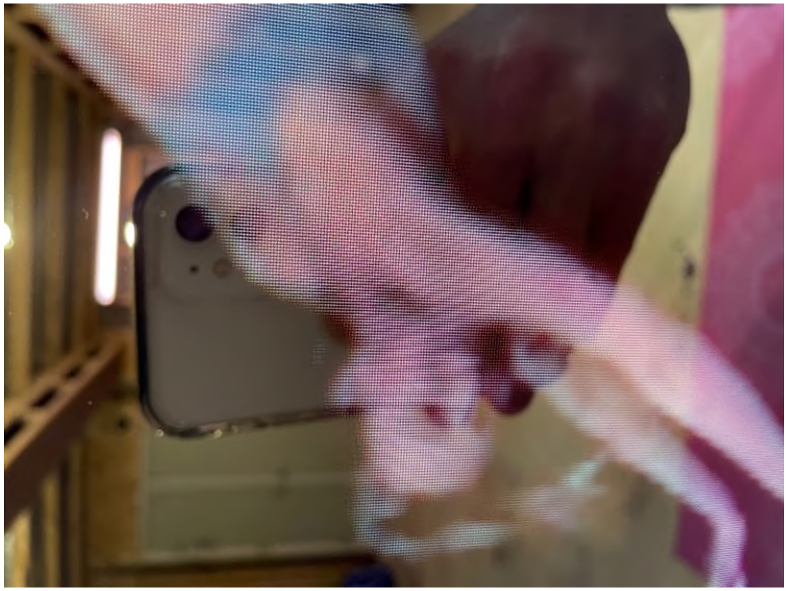
In order to demonstrate the bandwidth adaptation (“shift[ing] the playback quality”), the throttling feature of the Charles Proxy application was used to limit the Mirror Application’s bandwidth to approximate a slower speed, and then the throttling setting was removed.

When the bandwidth for the Mirror Application is reduced, the Mirror Application engages in adaptation to shift playback quality to a lower quality one of the different copies by requesting a lower bit rate version of the content at a subsequent time index, and when bandwidth for the Mirror Application is unconstrained, the Mirror Application engages in adaptation to shift playback quality to a higher quality one of the different copies in a similar way. This behavior demonstrates the automatic requesting dependent upon successive determinations by the media player to shift the playback quality. Recall from the discussion about the playlist.m3u8 file that the variant playlists and segments are stored in different directories, based upon the resolution and bandwidth.
<table>
<tr><th>Bandwidth</th><th>Playlist Filename</th></tr>
<tr><td>6434112</td><td>d1f65f45_1_4128/chunklist.m3u8</td></tr>
<tr><td>864048</td><td>d1f65f45_1_2728/chunklist.m3u8</td></tr>
</table>
</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		<table><tr><td>403824</td><td>d1f65f45_1_1728/chunklist.m3u8</td></tr><tr><td>367728</td><td>d1f65f45_1_1152/chunklist.m3u8</td></tr><tr><td>312832</td><td>d1f65f45_1_640/chunklist.m3u8</td></tr><tr><td>249664</td><td>d1f65f45_1_448/chunklist.m3u8</td></tr></table> The chosen resolutions of the Mirror Application can be determined based on which playlist and associated segments of the video are retrieved. A portion of the Charles Proxy sequence listing shows the requested and retrieved files while bandwidth was constrained and after the bandwidth was unconstrained is shown below. <table><tr><th>Method</th><th>Host</th><th>Path</th></tr><tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8</td></tr><tr><td colspan="3">…</td></tr><tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1277.ts</td></tr><tr><td colspan="3">…</td></tr><tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8</td></tr><tr><td colspan="3">…</td></tr><tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_640/3fyynadk/00000000/media_1278.ts</td></tr></table> When the Mirror Application operates at an unconstrained bandwidth, the Mirror Application transitions from a lower bitrate version of the video (e.g., the **249664 Bandwidth** version) to a higher bitrate version of the video (e.g., the **312832 Bandwidth** version). Between these two segments, the Mirror Application requests and receives a variant playlist file for the **312832 Bandwidth** version of the video ("/hls/live/268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8"). As demonstrated above, the Mirror Application requests and receives the lower resolution encoded files when the bandwidth is constrained and requests and receives the higher resolution encoded files when the bandwidth is unconstrained. In this way, the Mirror Application adapts subsequent segment requests based on successive determinations to shift the playback to a higher or lower quality encoding. On

Claim	Claim Limitation	Example Infringement Evidence
		information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above. For example, during a test of the Mirror Devices, the Mirror Devices successively determined to and shifted from a higher quality copy to a lower quality copy of the video when the bandwidth was constrained and then shifted back to a higher quality copy from the lower quality copy when the bandwidth was unconstrained, as shown below by the test capturing the shifting through multiple different copies of the video at varying qualities. First version: 

Claim	Claim Limitation	Example Infringement Evidence
		For the current test, when the bandwidth was limited, the Mirror Device subsequently shifted to a lower quality copy of the video. Second version: 

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>Third version:
</td></tr>
<tr><td></td><td>the automatically requesting including repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies,</td><td>The automatically requesting by the Mirror Application and Mirror Devices includes “repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the factor relates to the performance of the network.”

The repeated generation of a factor indicative of the current ability to sustain streaming of the video using the files from different ones of the copies is demonstrated by testing where bandwidth available to the Mirror Application and Mirror Devices is throttled and unthrottled and the playback quality automatically shifted in accordance with the player’s ability to sustain the stream under the bandwidth constraints, as was shown above.

In addition, the Mirror Devices run the Android operating system. ExoPlayer is a ubiquitous HLS video player for Android. ExoPlayer2 provide a DefaultBandwidthMeter class that “[e]stimates bandwidth by listening to data transfers.”</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
	wherein the factor relates to the performance of the network; and	https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/upstream/DefaultBandwidthMeter.html The ExoPlayer documentation further explains that "[t]he bandwidth estimate is calculated using a SlidingPercentile and is updated each time a transfer ends. The initial estimate is based on the current operator's network country code or the locale of the user, as well as the network connection type. This can be configured in the DefaultBandwidthMeter.Builder." *Id.* ExoPlayer's DefaultBandwidthMeter therefore repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the set of one or more factors relate to the performance of the network. The estimated bandwidth from the DefaultBandwidthMeter is then used by the AdaptiveTrackSelection.Factory class to determine whether to change to a higher or lower version of the stream. https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/trackselection/AdaptiveTrackSelection.Factory.html The AdaptiveTrackSelection.Factory class uses the following parameters to determine whether to switch: minDurationForQualityIncreaseMs - The minimum duration of buffered data required for the selected track to switch to one of higher quality. maxDurationForQualityDecreaseMs - The maximum duration of buffered data required for the selected track to switch to one of lower quality. minDurationToRetainAfterDiscardMs - When switching to a track of significantly higher quality, the selection may indicate that media already buffered at the lower quality can be discarded to speed up the switch. This is the minimum duration of media that must be retained at the lower quality.

Claim	Claim Limitation	Example Infringement Evidence
		bandwidthFraction - The fraction of the available bandwidth that the selection should consider available for use. Setting to a value less than 1 is recommended to account for inaccuracies in the bandwidth estimator. *Id*.
	making the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies determined sustainable at that time, wherein the making the successive determinations to shift comprises upshifting to a higher quality one of the different copies when the at least	The Mirror Application and Mirror Devices stream the video by "making the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies determined sustainable at that time." The successive determines also comprises "upshifting to a higher quality one of the different copies when the at least one factor is greater than a first threshold and downshifting to a lower quality one of the different copies when the at least one factor is less than a second threshold." The shifts between the higher and lower versions of the selected video demonstrates the determination to shift playback quality "based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies determined sustainable at that time." As noted above, the Mirror Application and Mirror Devices shift between playback quality based on a factor that includes, for example, bandwidth limitations to enable continuous playback of the video. Further, the Mirror Application and Mirror Devices request a lower quality content when bandwidth is constrained, as shown herein. Accordingly, the Mirror Application and Mirror Devices continue to request the highest quality content that is sustainable when doing so. Making the successive determinations to shift comprises upshifting to a higher quality one of the different copies when the at least one factor "is greater than a first threshold" and downshifting to a lower quality one of the different copies when the at least one factor "is less than second threshold." This is demonstrated at least by the Mirror Application shifting to the highest-quality version it is able to be based on bandwidth constraints. When the Mirror Application operates at unconstrained bandwidth, the Mirror Application transitions from a lower-bitrate version of the video (**249664 Bandwidth** version) to a higher-bitrate version of the

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td>one factor is greater than a first threshold and downshifting to a lower quality one of the different copies when the at least one factor is less than a second threshold; and</td><td>
video (367728 Bandwidth version). Between these two successive segments, the Mirror Application requests and receives a 367728 Bandwidth variant playlist file.

<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8</td></tr>
<tr><td colspan="3">..</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1279.ts</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1152/bmit701z/00000000/media_1278.ts</td></tr>
</table>

As demonstrated above, the Mirror Application requests and receives the higher resolution segment when operating at unconstrained bandwidth. Accordingly, the Mirror Application determined that the performance ratio exceeded a first threshold value prior to requesting the higher quality segments.

A portion of the Charles Proxy sequence listing below shows the requested and retrieved files prior to and just after bandwidth was constrained is shown below.

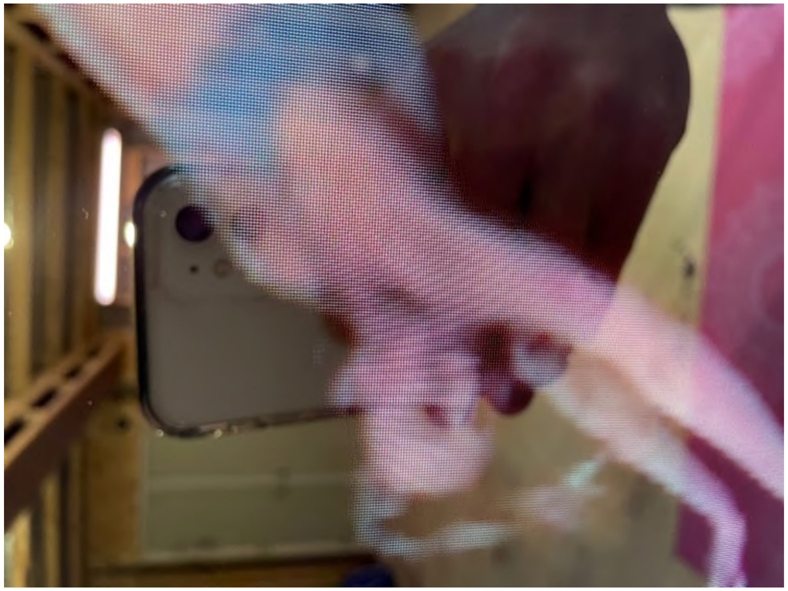
<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8</td></tr>
<tr><td colspan="3">..</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts</td></tr>
</table>
</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		When the Mirror Application operates at a constrained bandwidth, the Mirror Application transitions from a higher-bitrate version of the video (the **403824 Bandwidth** version) to a lower-bitrate version of the video (the **249664 Bandwidth** version). Between the two segments, the Mirror Application requests and receives a **249664 Bandwidth** variant playlist file of the video. As demonstrated above, the Mirror Application requests and receives the lower resolution encoded files while the bandwidth is constrained. Accordingly, the Mirror Application determines the performance ratio was less than a second threshold value prior to requesting lower quality segments. In operation, when full bandwidth is returned, the Mirror Application again requests and plays a higher bitrate segment of the video, which is in response the factor being greater than a threshold. Upon information and belief, the Mirror Devices operates in the same or substantially the same way. For example, when bandwidth is unconstrained, the Mirror Devices shift to the highest quality copy of the video that is sustainable.

Claim	Claim Limitation	Example Infringement Evidence
		First version:  For the current test, when the bandwidth was limited, the Mirror Device subsequently shifted to a lower quality copy of the video.

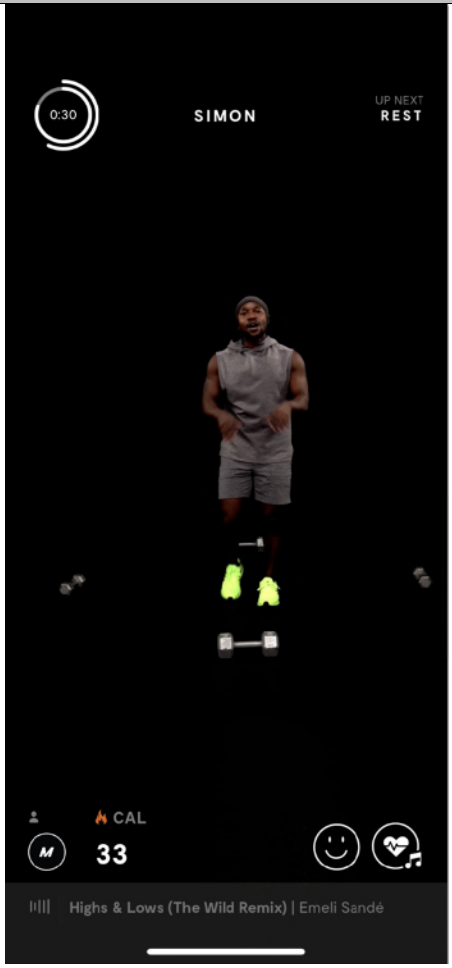
Claim	Claim Limitation	Example Infringement Evidence
		Second version: 

USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Third version:  When the bandwidth constraint was removed, the Mirror Device subsequently returned to a higher-bandwidth version shown above. The Mirror Devices run the Android operating system. ExoPlayer is a ubiquitous HLS video player for Android. ExoPlayer2 provide a DefaultBandwidthMeter class that "[e]stimates bandwidth by listening to data transfers." https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/upstream/DefaultBandwidthMeter.html. The ExoPlayer documentation further explains that "[t]he bandwidth estimate is calculated using a SlidingPercentile and is updated each time a transfer ends. The initial estimate is based on the current operator's network country code or the locale of the user, as well as the network connection type. This can be configured in the DefaultBandwidthMeter.Builder." *Id.*

Claim	Claim Limitation	Example Infringement Evidence
		ExoPlayer's DefaultBandwidthMeter therefore repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the set of one or more factors relate to the performance of the network. The estimated bandwidth from the DefaultBandwidthMeter is then used by the AdaptiveTrackSelection.Factory class to determine whether to change to a higher or lower version of the stream. https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/trackselection/AdaptiveTrackSelection.Factory.html The AdaptiveTrackSelection.Factory class uses the following parameters to determine whether to switch: minDurationForQualityIncreaseMs - The minimum duration of buffered data required for the selected track to switch to one of higher quality. maxDurationForQualityDecreaseMs - The maximum duration of buffered data required for the selected track to switch to one of lower quality. minDurationToRetainAfterDiscardMs - When switching to a track of significantly higher quality, the selection may indicate that media already buffered at the lower quality can be discarded to speed up the switch. This is the minimum duration of media that must be retained at the lower quality. bandwidthFraction - The fraction of the available bandwidth that the selection should consider available for use. Setting to a value less than 1 is recommended to account for inaccuracies in the bandwidth estimator. *Id*. ExoPlayer therefore makes the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the files of the highest quality one of the copies determined sustainable at that time so that the media player upshifts to a higher quality one of the different copies when the factor is greater than a first threshold and downshifts to a lower quality one of the different copies when the factor is less than a second threshold.

Claim	Claim Limitation	Example Infringement Evidence
	presenting the video by playing back the requested media files with the media player on the end user station in order of ascending playback time.	As shown above, the Mirror Application and Mirror Devices stream the video by receiving the playlist file that lists the .ts file segments in order of ascending playback time and the Mirror Application requests those same .ts files in order of ascending playback time. The Mirror Application and Mirror Devices "present[] the video by playing back the requested media files with the media player on the end user station in order of ascending playback time." The Mirror Application and Mirror Devices playback the requested .ts files in order of ascending playback time after they are retrieved. For example, the media player of the Mirror Application includes a timer or timeline indicating that the media files are played back in order of ascending playback time in accordance with the timeline.

Claim	Claim Limitation	Example Infringement Evidence
		 For example, the media player of the Mirror Devices includes a timer or timeline indicating that the media files are played back in order of ascending playback time in accordance with the timeline.

USP 9,407,564 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		

EXHIBIT B

US010469554B2

(12) **United States Patent**
Brueck et al.

(10) **Patent No.: US 10,469,554 B2**
(45) **Date of Patent: *Nov. 5, 2019**

(54) **APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING**

(71) Applicant: **DISH Technologies L.L.C.**, Englewood, CO (US)

(72) Inventors: **David F. Brueck**, Saratoga Springs, UT (US); **Mark B. Hurst**, Cedar Hills, UT (US); **R. Drew Major**, Orem, UT (US)

(73) Assignee: **DISH Technologies L.L.C.**, Englewood, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **16/252,188**

(22) Filed: **Jan. 18, 2019**

(65) **Prior Publication Data**

US 2019/0158560 A1 May 23, 2019

Related U.S. Application Data

(63) Continuation of application No. 16/004,056, filed on Jun. 8, 2018, which is a continuation of application (Continued)

(51) **Int. Cl.**
H04L 29/06 (2006.01)
H04L 12/927 (2013.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04L 65/607*** (2013.01); ***G06F 16/183*** (2019.01); ***G06F 16/71*** (2019.01);
(Continued)

(58) **Field of Classification Search**
CPC .. H04N 19/34; H04N 19/40; H04N 21/23427; H04N 21/2662;
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,535,355	A	8/1985	Am et al.
5,168,356	A	12/1992	Acampora et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CA	2466482	A1	5/2003
EP	0919952	A1	6/1999

(Continued)

OTHER PUBLICATIONS

Fujisawa, Hiroshi et al. "Implementaton of Efficient Access Mechanism for Multiple Mirror-Servers" IPSJ SIG Technical Report, vol. 2004, No. 9 (2004-DPS-116), Jan. 30, 2004, Information Processing Society of Japan, pp. 37-42.

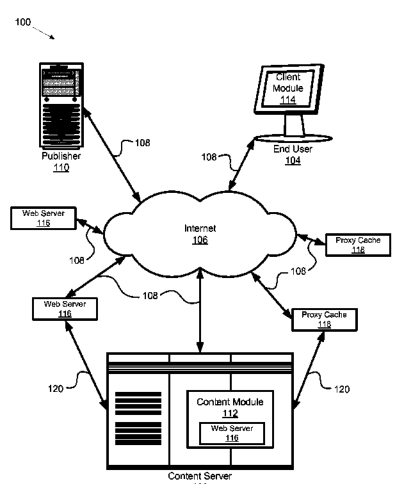
(Continued)

Primary Examiner — Chirag R Patel
(74) *Attorney, Agent, or Firm* — Lorenz & Kopf LLP

(57) **ABSTRACT**

An apparatus for multi-bitrate content streaming includes a receiving module configured to capture media content, a streamlet module configured to segment the media content and generate a plurality of streamlets, and an encoding module configured to generate a set of streamlets. The system includes the apparatus, wherein the set of streamlets comprises a plurality of streamlets having identical time indices and durations, and each streamlet of the set of streamlets having a unique bitrate, and wherein the encoding module comprises a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid. A method includes receiving media content, segmenting the media content and generating a plurality of streamlets, and generating a set of streamlets.

30 Claims, 11 Drawing Sheets



Related U.S. Application Data

No. 15/414,025, filed on Jan. 24, 2017, now Pat. No. 9,998,516, which is a continuation of application No. 14/719,122, filed on May 21, 2015, now Pat. No. 9,571,551, which is a continuation of application No. 14/106,051, filed on Dec. 13, 2013, now Pat. No. 9,071,668, which is a continuation of application No. 13/617,114, filed on Sep. 14, 2012, now Pat. No. 8,612,624, which is a continuation of application No. 12/906,940, filed on Oct. 18, 2010, now Pat. No. 8,402,156, which is a continuation of application No. 11/673,483, filed on Feb. 9, 2007, now Pat. No. 7,818,444, which is a continuation-in-part of application No. 11/116,783, filed on Apr. 28, 2005, now Pat. No. 8,868,772.

(60) Provisional application No. 60/566,831, filed on Apr. 30, 2004.

(51) **Int. Cl.**

H04L 12/801	(2013.01)
G06F 16/71	(2019.01)
G06F 16/182	(2019.01)
H04N 7/24	(2011.01)
H04N 21/2343	(2011.01)
H04N 21/433	(2011.01)
H04N 21/84	(2011.01)
H04N 21/845	(2011.01)
H04L 29/08	(2006.01)
H04N 21/2662	(2011.01)

(52) **U.S. Cl.**

CPC ***H04L 29/06027*** (2013.01); ***H04L 47/12*** (2013.01); ***H04L 47/801*** (2013.01); ***H04L 65/1069*** (2013.01); ***H04L 65/4069*** (2013.01); ***H04L 65/608*** (2013.01); ***H04L 65/80*** (2013.01); ***H04L 67/02*** (2013.01); ***H04L 67/2842*** (2013.01); ***H04L 67/32*** (2013.01); ***H04N 7/24*** (2013.01); ***H04N 21/23439*** (2013.01); ***H04N 21/2662*** (2013.01); ***H04N 21/4331*** (2013.01); ***H04N 21/84*** (2013.01); ***H04N 21/8456*** (2013.01)

(58) **Field of Classification Search**

CPC H04N 21/234327; H04N 21/2393; H04L 65/80; H04L 67/2842; H04L 65/4069; H04L 65/607; H04L 65/608

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,267,334 A 11/1993 Normille et al.
5,404,446 A 4/1995 Bowater et al.
5,687,095 A 11/1997 Haskell et al.
5,732,183 A 3/1998 Sugiyama
5,768,527 A 6/1998 Zhu et al.
5,812,786 A * 9/1998 Seazholtz H04M 11/062 370/465
5,841,432 A 11/1998 Carmel et al.
5,953,506 A 9/1999 Kalra et al.
6,091,775 A 7/2000 Hibi et al.
6,091,777 A 7/2000 Guetz et al.
6,122,660 A 9/2000 Baransky et al.
6,185,736 B1 2/2001 Ueno
6,195,680 B1 2/2001 Goldszmidt et al.
6,366,614 B1 4/2002 Pian et al.
6,374,289 B2 4/2002 Delaney et al.
6,389,473 B1 5/2002 Carmel et al.
6,449,719 B1 9/2002 Baker
6,486,803 B1 11/2002 Luby et al.
6,490,627 B1 12/2002 Kalra et al.
6,496,980 B1 * 12/2002 Tillman H04N 7/17318 348/E7.071
6,510,553 B1 1/2003 Hazra
6,574,591 B1 6/2003 Kleiman et al.
6,604,118 B2 8/2003 Klleiman et al.
6,618,752 B1 9/2003 Moore et al.
6,708,213 B1 3/2004 Bommaiah et al.
6,721,723 B1 4/2004 Gibson et al.
6,731,600 B1 5/2004 Patel et al.
6,757,796 B1 6/2004 Hofmann
6,760,772 B2 7/2004 Zou et al.
6,795,863 B1 9/2004 Doty, Jr.
6,845,107 B1 1/2005 Kitazawa et al.
6,850,965 B2 2/2005 Allen
6,859,839 B1 2/2005 Zahorjan et al.
6,874,015 B2 3/2005 Kaminsky et al.
6,968,387 B2 11/2005 Lanphear
6,976,090 B2 12/2005 Ben-Shaul et al.
7,054,365 B2 5/2006 Kim et al.
7,054,774 B2 5/2006 Batterberry et al.
7,054,911 B1 5/2006 Lango et al.
7,075,986 B2 7/2006 Girod et al.
7,093,001 B2 8/2006 Yang et al.
7,096,271 B1 8/2006 Omoigui et al.
7,099,954 B2 8/2006 Li et al.
7,116,894 B1 10/2006 Chatterton
7,174,385 B2 2/2007 Li
7,194,549 B1 3/2007 Lee et al.
7,240,100 B1 7/2007 Wein et al.
7,260,640 B1 8/2007 Kramer et al.
7,274,740 B2 9/2007 van Beek et al.
7,295,520 B2 11/2007 Lee et al.
7,310,678 B2 12/2007 Gunaseelan et al.
7,325,073 B2 1/2008 Shao et al.
7,328,243 B2 2/2008 Yaeger et al.
7,330,908 B2 2/2008 Jungek
7,334,044 B1 2/2008 Allen
7,349,358 B2 3/2008 Hennessey et al.
7,349,976 B1 3/2008 Glaser et al.
7,369,610 B2 * 5/2008 Xu H04N 21/2662 375/240.08
7,376,747 B2 5/2008 Hartop
7,391,717 B2 6/2008 Kiemets et al.
7,408,984 B2 8/2008 Lu et al.
7,412,531 B1 8/2008 Lango et al.
7,477,688 B1 1/2009 Zhang et al.
7,523,181 B2 4/2009 Swildens et al.
7,536,469 B2 5/2009 Chou et al.
7,546,355 B2 6/2009 Kalnitsky
7,558,869 B2 7/2009 Leon et al.
7,577,750 B2 8/2009 Shen et al.
7,593,333 B2 9/2009 Li et al.
7,599,307 B2 10/2009 Seckni et al.
7,609,652 B2 10/2009 Kellerer et al.
7,653,735 B2 1/2010 Mandato et al.
7,707,303 B2 4/2010 Albers et al.
7,719,985 B2 5/2010 Lee et al.
7,760,801 B2 7/2010 Ghanbari et al.
7,779,135 B2 8/2010 Hudson et al.
7,788,395 B2 8/2010 Bowra et al.
7,797,439 B2 9/2010 Cherkasova et al.
7,817,985 B2 10/2010 Moon
7,818,444 B2 10/2010 Brueck et al.
7,925,781 B1 4/2011 Chan et al.
7,934,159 B1 * 4/2011 Rahman H04N 21/4825 715/716
8,036,265 B1 10/2011 Reynolds et al.
8,370,514 B2 2/2013 Hurst et al.
8,402,156 B2 3/2013 Brueck et al.
8,521,836 B2 8/2013 Kewalramani et al.
8,612,624 B2 12/2013 Brueck et al.
8,683,066 B2 3/2014 Hurst et al.
8,686,066 B2 4/2014 Kwampian et al.
8,868,772 B2 10/2014 Major et al.
8,880,721 B2 11/2014 Hurst et al.
9,344,496 B2 5/2016 Hurst et al.
9,462,074 B2 10/2016 Guo et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2001/0013128 A1	8/2001	Hagai et al.	
2001/0047423 A1	11/2001	Shao et al.	
2002/0029274 A1	3/2002	Allen	
2002/0073167 A1	6/2002	Powell et al.	
2002/0091840 A1	7/2002	Pulier et al.	
2002/0097750 A1	7/2002	Gunaseelan et al.	
2002/0131496 A1	9/2002	Vasudevan et al.	
2002/0144276 A1	10/2002	Radford et al.	
2002/0152317 A1	10/2002	Wang et al.	
2002/0152318 A1	10/2002	Menon et al.	
2002/0156912 A1	10/2002	Hurst et al.	
2002/0161898 A1	10/2002	Hartop et al.	
2002/0161908 A1	10/2002	Benitez et al.	
2002/0161911 A1	10/2002	Pinckney, III et al.	
2002/0169926 A1	11/2002	Pinckney, III et al.	
2002/0174434 A1	11/2002	Lee et al.	
2002/0176418 A1	11/2002	Hunt et al.	
2002/0178330 A1	11/2002	Schlowsky-Fischer et al.	
2002/0188745 A1	12/2002	Hughes et al.	
2003/0005455 A1	1/2003	Bowers	
2003/0014684 A1	1/2003	Kashyap	
2003/0018966 A1	1/2003	Cook et al.	
2003/0021166 A1	1/2003	Soloff	
2003/0021282 A1	1/2003	Hospodor	
2003/0023982 A1*	1/2003	Lee	H04N 21/23432 725/116
2003/0055995 A1	3/2003	Ala Honkola	
2003/0065803 A1	4/2003	Heuvelman	
2003/0067872 A1	4/2003	Harrell et al.	
2003/0081582 A1	5/2003	Jain et al.	
2003/0093790 A1	5/2003	Logan et al.	
2003/0103571 A1*	6/2003	Meehan	H04N 21/23432 375/240.27
2003/0107994 A1	6/2003	Jacobs et al.	
2003/0135631 A1	7/2003	Li et al.	
2003/0140159 A1	7/2003	Campbell et al.	
2003/0151753 A1	8/2003	Li et al.	
2003/0152036 A1	8/2003	Quigg Brown et al.	
2003/0154239 A1	8/2003	Davis et al.	
2003/0195977 A1	10/2003	Liu et al.	
2003/0204519 A1	10/2003	Sirivara et al.	
2003/0204602 A1	10/2003	Hudson et al.	
2003/0236904 A1	12/2003	Walpole et al.	
2004/0003101 A1	1/2004	Roth et al.	
2004/0010613 A1	1/2004	Apostolopoulos et al.	
2004/0030547 A1	2/2004	Leaning et al.	
2004/0030599 A1	2/2004	Sie et al.	
2004/0030797 A1	2/2004	Akinlar et al.	
2004/0031054 A1	2/2004	Dankworth et al.	
2004/0049780 A1	3/2004	Gee	
2004/0054551 A1	3/2004	Ausubel et al.	
2004/0071209 A1	4/2004	Burg et al.	
2004/0083283 A1	4/2004	Sundaram et al.	
2004/0093420 A1	5/2004	Gamble	
2004/0103444 A1	5/2004	Weinberg et al.	
2004/0117427 A1	6/2004	Allen et al.	
2004/0143672 A1	7/2004	Padmanabham et al.	
2004/0168052 A1	8/2004	Clisham et al.	
2004/0170392 A1	9/2004	Lu et al.	
2004/0179032 A1	9/2004	Huang	
2004/0199655 A1	10/2004	Davies et al.	
2004/0220926 A1	11/2004	Lamkin et al.	
2004/0221088 A1	11/2004	Lisitsa et al.	
2004/0260701 A1	12/2004	Lehikoinen et al.	
2004/0267956 A1	12/2004	Leon et al.	
2005/0015509 A1	1/2005	Sitaraman	
2005/0033855 A1	2/2005	Moradi et al.	
2005/0055425 A1*	3/2005	Lango	H04L 29/06027 709/219
2005/0066063 A1	3/2005	Grigorovitch et al.	
2005/0076136 A1	4/2005	Cho et al.	
2005/0084166 A1	4/2005	Bonch et al.	
2005/0108414 A1	5/2005	Taylor et al.	
2005/0120107 A1	6/2005	Kagan et al.	
2005/0123058 A1	6/2005	Greenbaum et al.	
2005/0185578 A1	8/2005	Padmanabham et al.	
2005/0188051 A1	8/2005	Sneh	
2005/0204046 A1	9/2005	Watanabe	
2005/0251832 A1	11/2005	Chiueh	
2005/0262257 A1	11/2005	Major et al.	
2006/0010003 A1	1/2006	Kruse	
2006/0059223 A1	3/2006	Klemets et al.	
2006/0075446 A1	4/2006	Klemets et al.	
2006/0080718 A1	4/2006	Gray et al.	
2006/0130118 A1	6/2006	Damm	
2006/0133809 A1	6/2006	Chow et al.	
2006/0165166 A1	7/2006	Chou et al.	
2006/0168290 A1	7/2006	Doron	
2006/0168295 A1	7/2006	Batterberry et al.	
2006/0206246 A1	9/2006	Walker	
2006/0236219 A1	10/2006	Grigorovitch et al.	
2006/0277564 A1	12/2006	Jarman	
2007/0024705 A1	2/2007	Richter et al.	
2007/0030833 A1	2/2007	Pirzada et al.	
2007/0067480 A1	3/2007	Beek et al.	
2007/0079325 A1	4/2007	de Heer	
2007/0094405 A1	4/2007	Zhang	
2007/0204310 A1	8/2007	Hua et al.	
2007/0280255 A1	12/2007	Tsang et al.	
2008/0028428 A1	1/2008	Jeong et al.	
2008/0037527 A1	2/2008	Chan et al.	
2008/0046939 A1	2/2008	Lu et al.	
2008/0056373 A1	3/2008	Newlin et al.	
2008/0104647 A1	5/2008	Hannuksela	
2008/0120330 A1	5/2008	Reed et al.	
2008/0120342 A1	5/2008	Reed et al.	
2008/0133766 A1	6/2008	Luo	
2008/0162713 A1	7/2008	Bowra et al.	
2008/0184688 A1	8/2008	Daly et al.	
2008/0195744 A1	8/2008	Bowra et al.	
2008/0205291 A1	8/2008	Li et al.	
2008/0219151 A1	9/2008	Ma et al.	
2008/0222235 A1	9/2008	Hurst et al.	
2008/0263180 A1	10/2008	Hurst et al.	
2008/0281803 A1	11/2008	Gentric	
2009/0043906 A1	2/2009	Hurst et al.	
2009/0055471 A1	2/2009	Kozat et al.	
2009/0055547 A1	2/2009	Hudson et al.	
2009/0210549 A1	8/2009	Hudson et al.	
2010/0098103 A1	4/2010	Xiong et al.	
2010/0262711 A1	10/2010	Bouazizi	
2011/0307545 A1	12/2011	Bouazizi	
2015/0058496 A1	2/2015	Hurst et al.	

FOREIGN PATENT DOCUMENTS

EP	1202487 A2	5/2002
EP	1298931 A2	4/2003
EP	1395014 A1	3/2004
EP	1670256 A2	6/2006
EP	1777969	4/2007
GB	2367219 A	3/2002
JP	2000-201343	7/2000
JP	200192752	4/2001
JP	2011004225 A	1/2011
WO	2001067264 A1	9/2001
WO	2004025405 A2	3/2004
WO	2006010113 A2	1/2006

OTHER PUBLICATIONS

Liu, Jiangchuan et al. "Adaptive Video Multicast Over the Internet" IEEE Computer Society, 2003.

"The meaning of performance factor—English-Japanese Weblio Dictionary", [online], Feb. 24, 2012, [searched on Feb. 24, 2012], the Internet <URL:http://ejje.weblio.jp/content/performance+factor>.

Tsuru, et al. "Recent evolution of the Internet measurement and inference techniques", IEICE Technical Report, vol. 103, No. 123, pp. 37-42, Jun. 12, 2003.

Rejaie, Reza et al. "Architectural Considerations for Playback of Quality Adaptive Video OVer the Internet" University of Southern California, Information Sciences Institute, 1998.

(56) **References Cited**

OTHER PUBLICATIONS

Roy, Sumit et al. "A System Architecture for Managing Mobile Streaming Media Services" Streaming Media Systems Group, Hewlett-Packard Laboratories, 2003.
Xu, Dongyan et al. "On Peer-to-Peer Media Streaming" Department of Computer Sciences, Purdue University, 2002.
Kozamerink, Franc "Media Streaming Over the Internet—An Over of Delivery Technologies" EBU Technical Review, Oct. 2002.
Lienhart, Rainer et al. "Challenges in Distributed Video Management and Delivery" Intel Corporation, EECS Dept., UC Berkeley, 2000-2002.
Zhang, Xinyan et al. "CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming" IEEE 2005.
Guo, Yang "DirectStream: A Directory-Based Peer-to-Peer Video Streaming Service" LexisNexis, Elsevier B.V. 2007.
Krasic et al., Quality-Adaptive Media Streaming by Priority Drop, Oregon Graduate Institute, 2001.
Krasic et al., QoS Scalability for Streamed Media Delivery, Oregon Graduate Institute School of Science & Engineering Technical Report CSE 99-011, Sep. 1999.
Huang et al., Adaptive Live Video Streaming by Priority Drop, Portland State University PDXScholar, Jul. 21, 2003.
Walpole et al, A Player for Adapctive MPEG Video Streaming Over the Internet, Oregon Graduate Institute of Science and Technology, Oct. 25, 2012.
Albanese, Andrew et al. "Priority Encoding Transmission", TR-94-039, Aug. 1994, 36 pgs, International Computer Science Institute, Berkeley, CA.
Birney, Bill "Intelligent Streaming", May 2003, Microsoft.
Goyal, Vivek K. "Multiple Description Coding: Compression Meets the Network," Sep. 2001, pp. 74-93, IEEE Signal Processing Magazine.
ON2 Technologies, Inc. "TrueMotion VP7 Video Codec" White Paper, Document Version 1.0, Jan. 10, 2005.
Pathan, Al-Mukaddim et al. "A Taxonomy and Survey of Content Delivery Networks" Australia, Feb. 2007, available at http://www.gridbus.org/reports/CDN-Taxonomy.pdf.
Puri, Rohit et al. "Multiple Description Source Coding Using Forward Error Correction Codes," Oct. 1999, 5 pgs., Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA.
Wicker, Stephen B. "Error Control Systems for Digital Communication and Storage," Prentice-Hall, Inc., New Jersey, USA, 1995, parts 1-6.
Liu, Jiangchuan et al. "Opportunities and Challenged of Peer-to-Peer Internet Video Broadcast," School of Computing Science, Simon Fraser University, British Columbia, Canada.
Clement, B. "Move Networks closes $11.3 Million on First Round VC Funding," Page One PR, Move Networks, Inc. Press Releases, Feb. 7, 2007, http://www.move.tv/press/press20070201.html.
Move Networks, Inc. "The Next Generation Video Publishing System," Apr. 11, 2007; http://www.movenetworks.com/wp-content/uploads/move-networks-publishing-system.pdf.
Yoshimura, Takeshi et al. "Mobile Streaming Media CDN Enabled by Dynamic SMIL", NTT DoCoMo, Multimedia Laboratories and Hewlett-Packard Laboratories,dated May 7-11, 2002, ACM 1-58113-449-5/02/0005; http://www2002.org/CDROM/refereed/515/.
Nguyen, T. et al., Multiple Sender Distributed Video Streaming, IEEE Transactinos on Multimedia, IEEE Service Center, Piscataway, NJ, US, vol. 6, No. 2, Apr. 1, 2004, pp. 315-326, XP011109142, ISSN: 1520-9210, DOI: 10.1109/TMM,2003.822790.
Roy, S., et al., "Architecture of a Modular Streaming Media Server for Content Delivery Networks," 2002 IEEE. Published in the 2003 International Conference on Multimedia and Expo ICME 2003.
Bommaiah, E., et al., "Design and Implementation of a Caching System for Streaming Media over the Internet," 2000 IEEE. Published in RTAS '00 Proceedings of the Sixth IEEE Real Time Technology and Applications Symposium (RTAS 2000), p. 111.

* cited by examiner

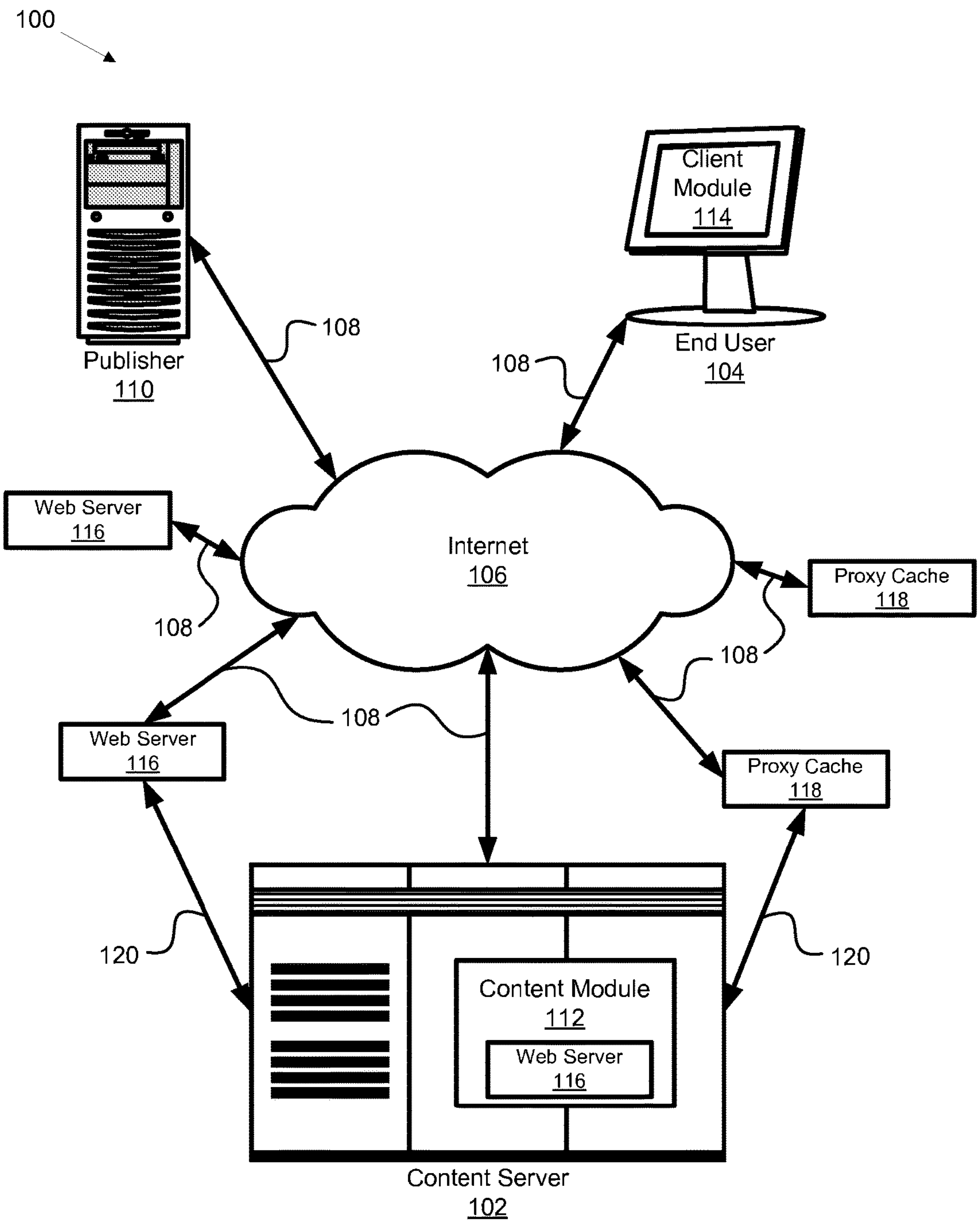

100
Publisher
110
Client
Module
114
End User
104
108
108
Web Server
116
Internet
106
Proxy Cache
118
108
108
108
Web Server
116
Proxy Cache
118
120
120
Content Module
112
Web Server
116
Content Server
102

FIG. 1

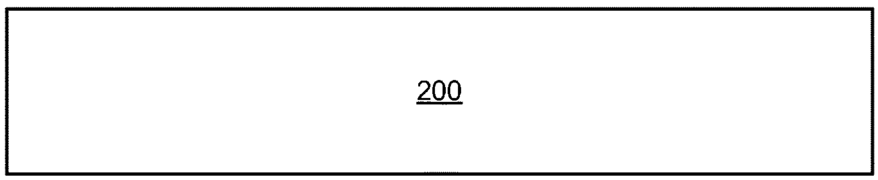


FIG. 2a

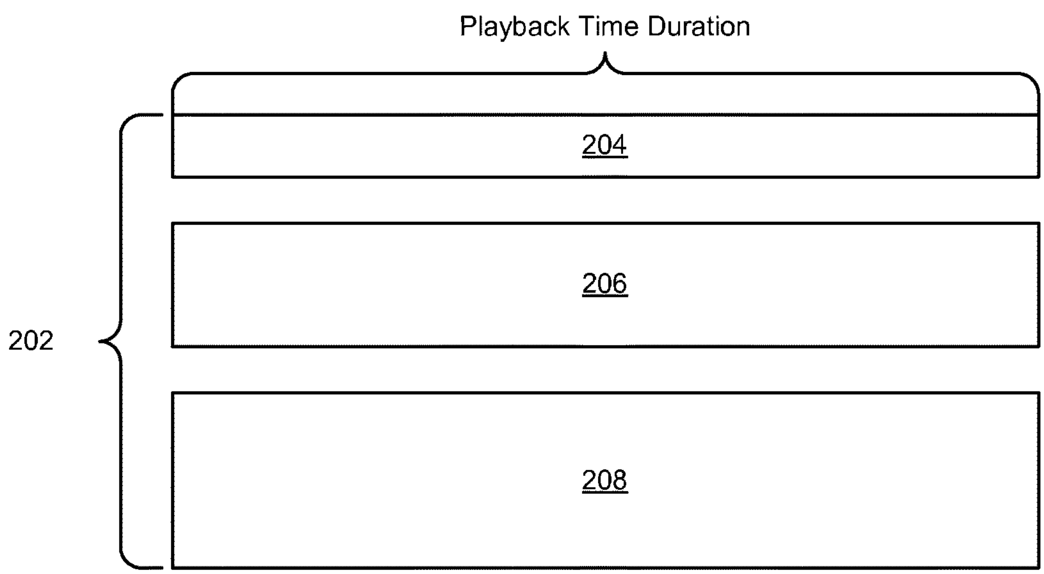


FIG. 2b

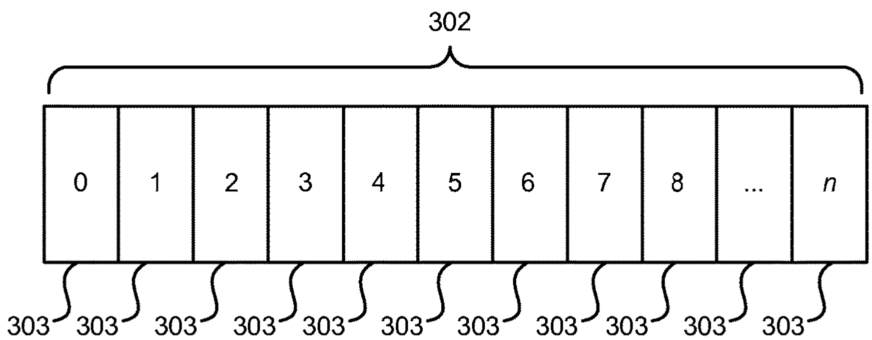

302
0
1
2
3
4
5
6
7
8
...
n
303
303
303
303
303
303
303
303
303
303
303

FIG. 3a

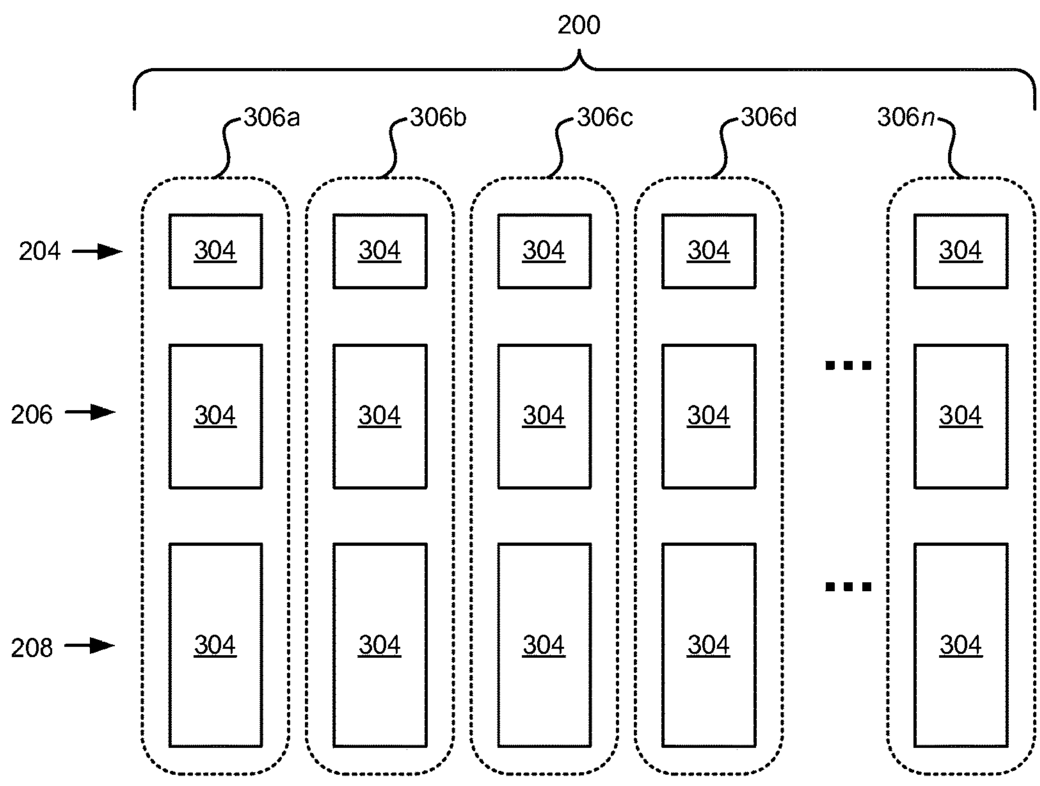
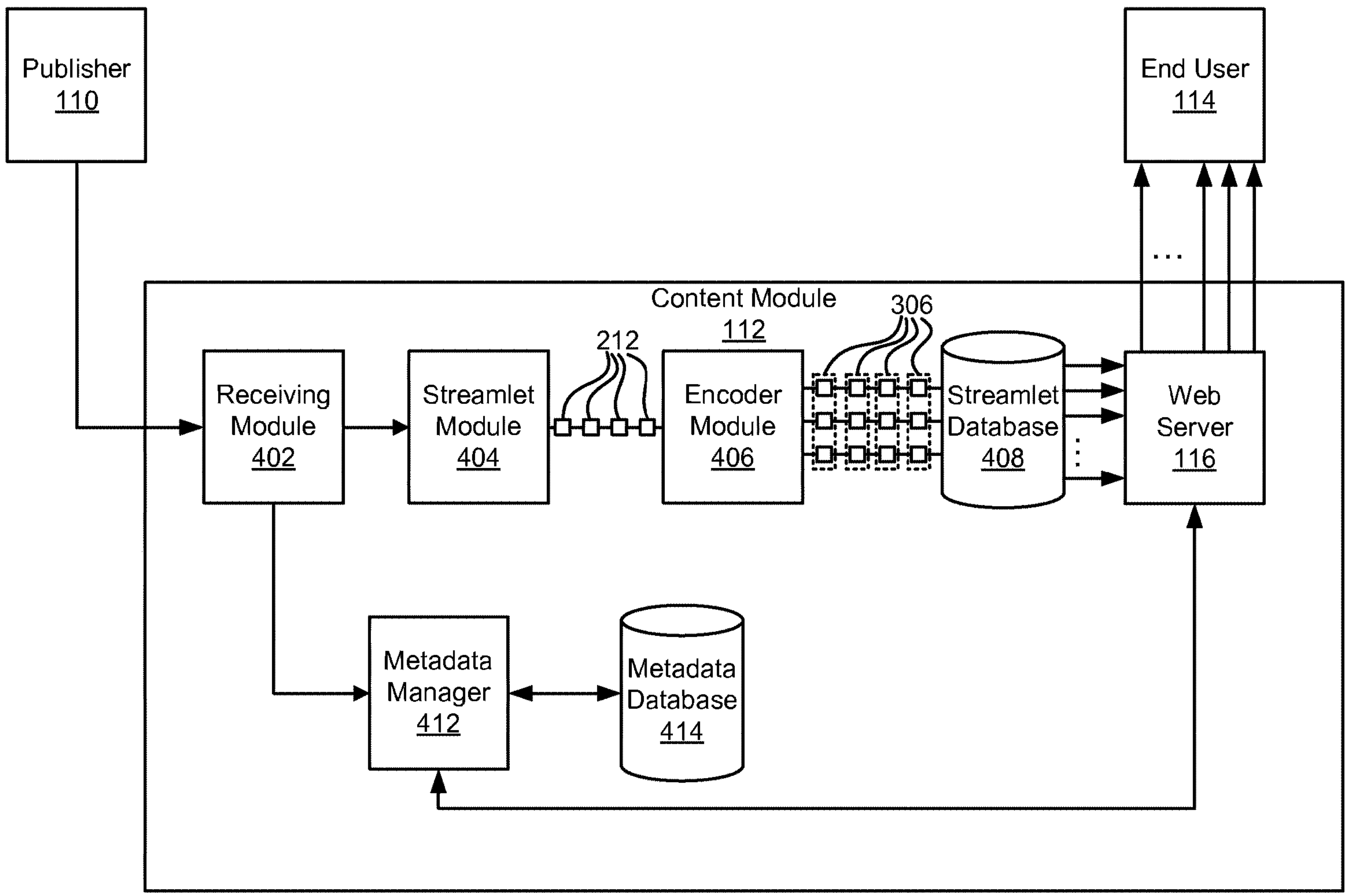

200
306a
306b
306c
306d
306n
204
206
208
304

FIG. 3b



Publisher
110
End User
114
Content Module
112
306
212
Receiving Module
402
Streamlet Module
404
Encoder Module
406
Streamlet Database
408
Web Server
116
Metadata Manager
412
Metadata Database
414

FIG. 4

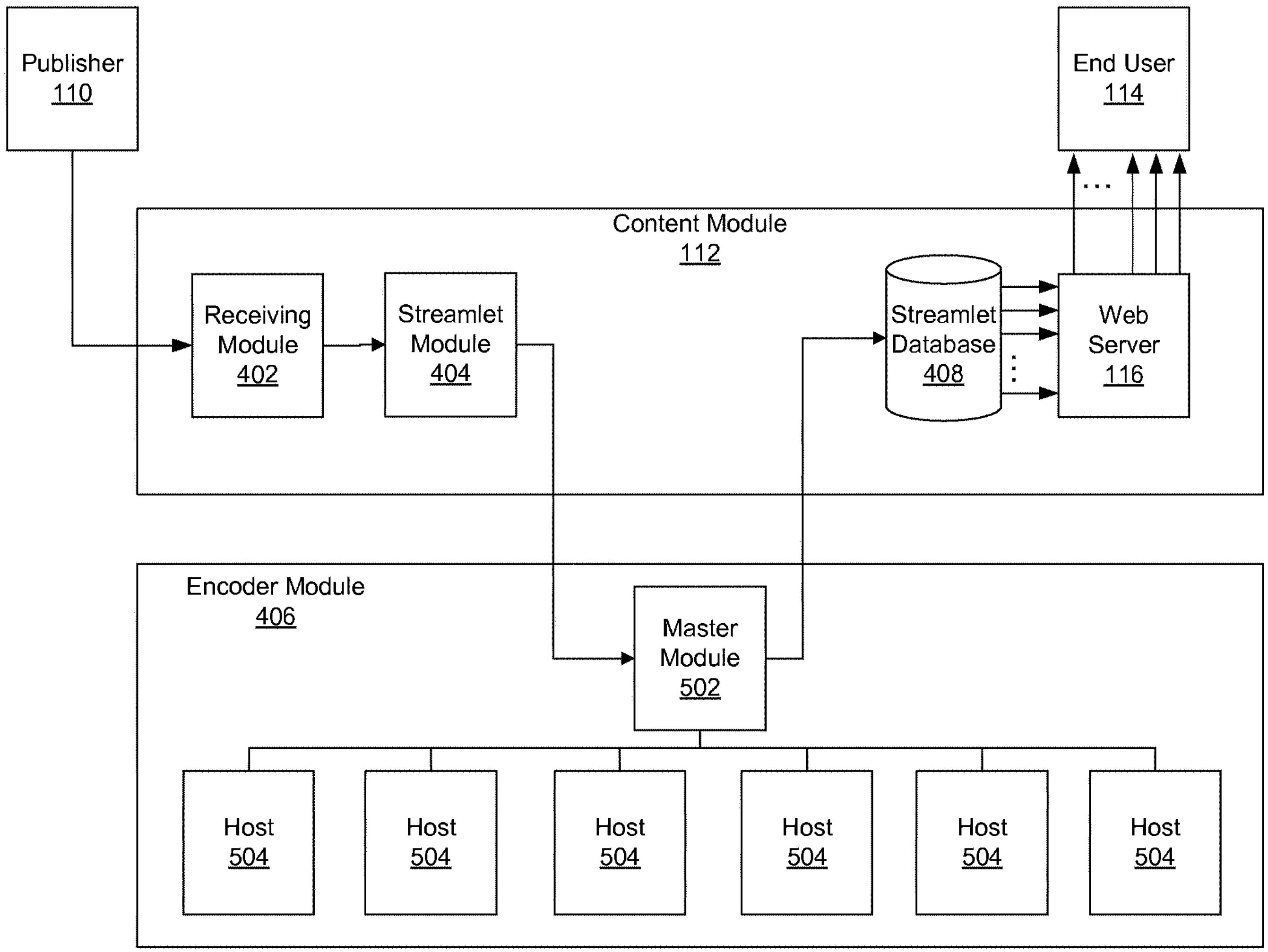

Publisher
110
End User
114
Content Module
112
Receiving Module
402
Streamlet Module
404
Streamlet Database
408
Web Server
116
Encoder Module
406
Master Module
502
Host
504
Host
504
Host
504
Host
504
Host
504
Host
504

FIG. 5a

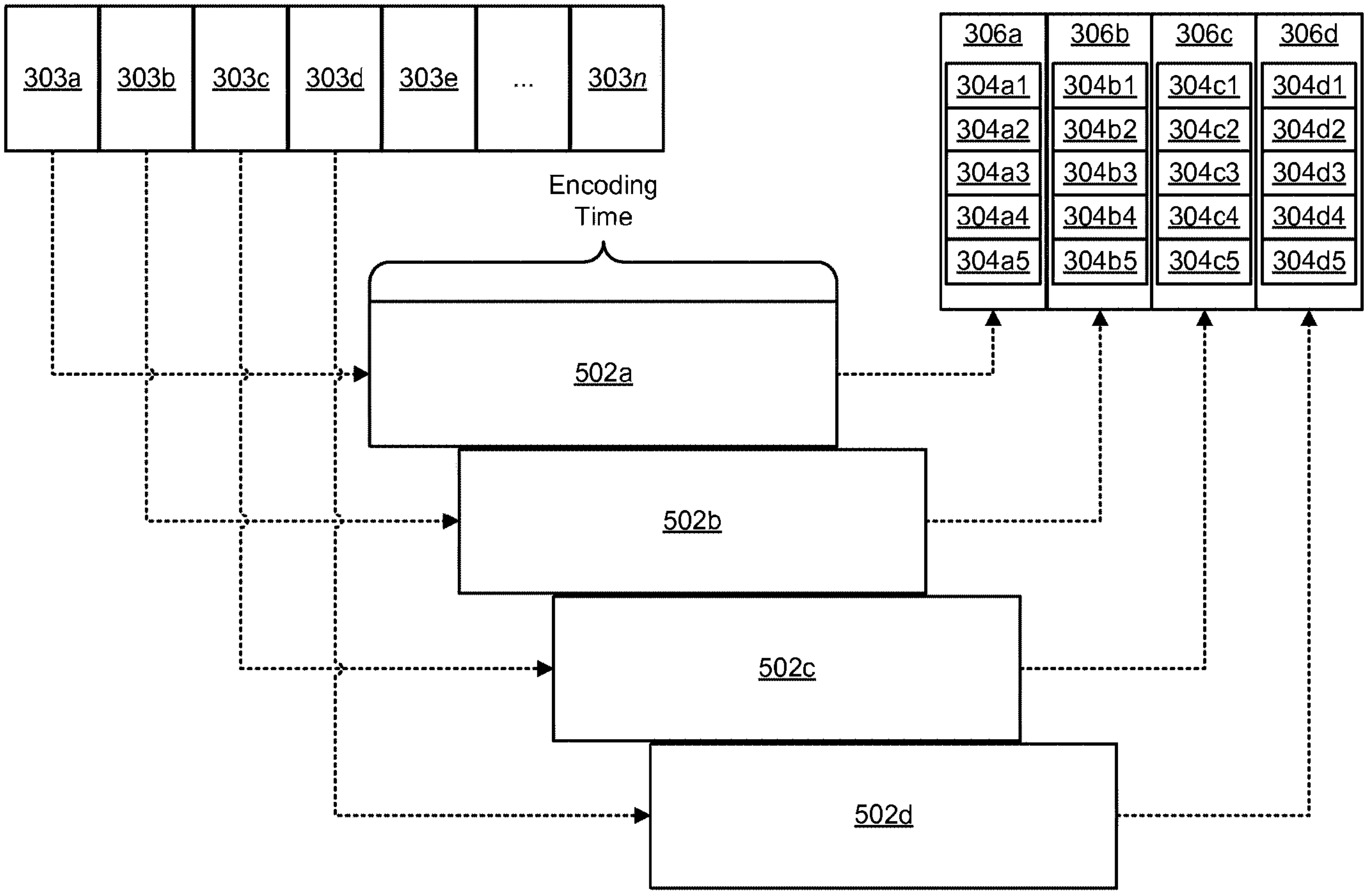
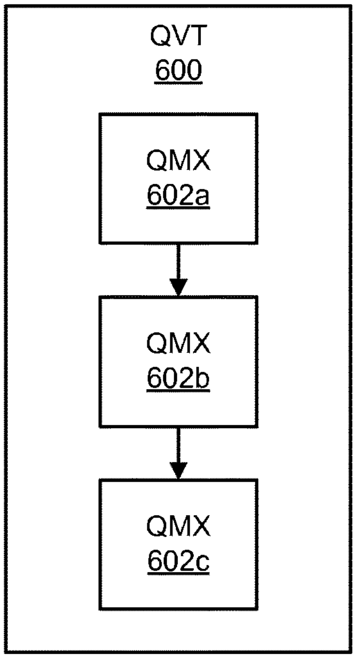

303a
303b
303c
303d
303e
...
303n
Encoding Time
502a
502b
502c
502d
306a
306b
306c
306d
304a1
304a2
304a3
304a4
304a5
304b1
304b2
304b3
304b4
304b5
304c1
304c2
304c3
304c4
304c5
304d1
304d2
304d3
304d4
304d5

FIG. 5b



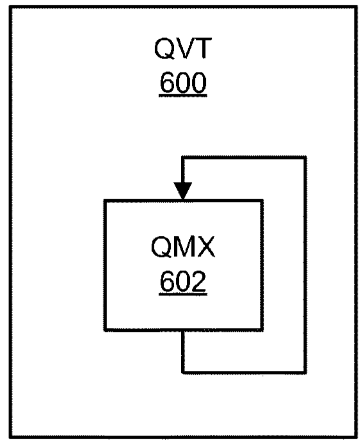


FIG. 6b

FIG. 6a

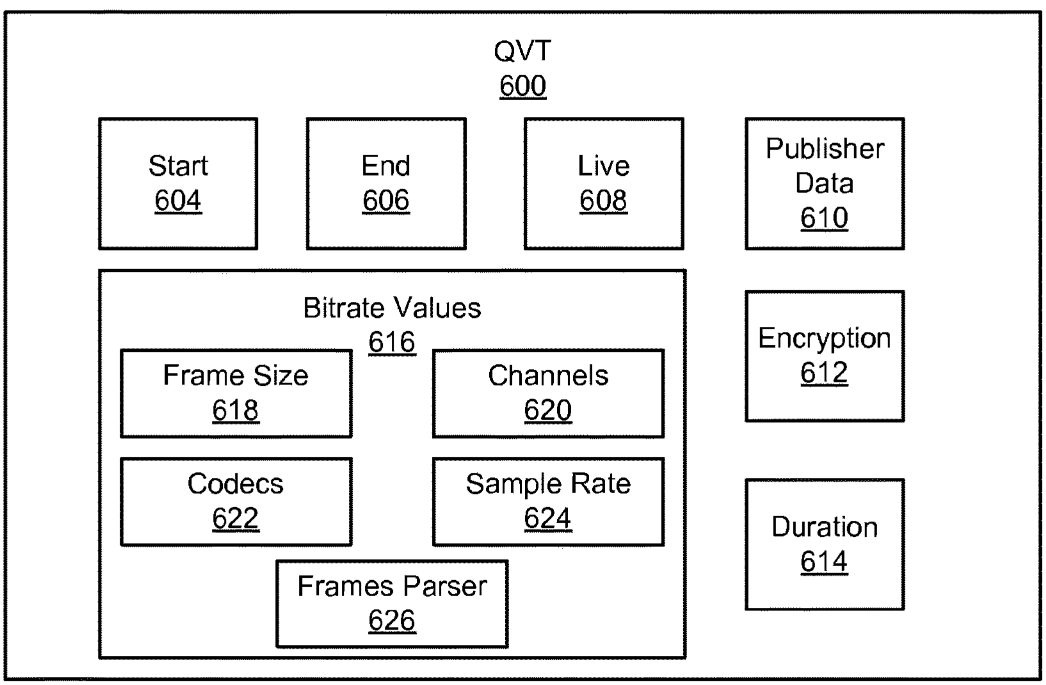


FIG. 6c

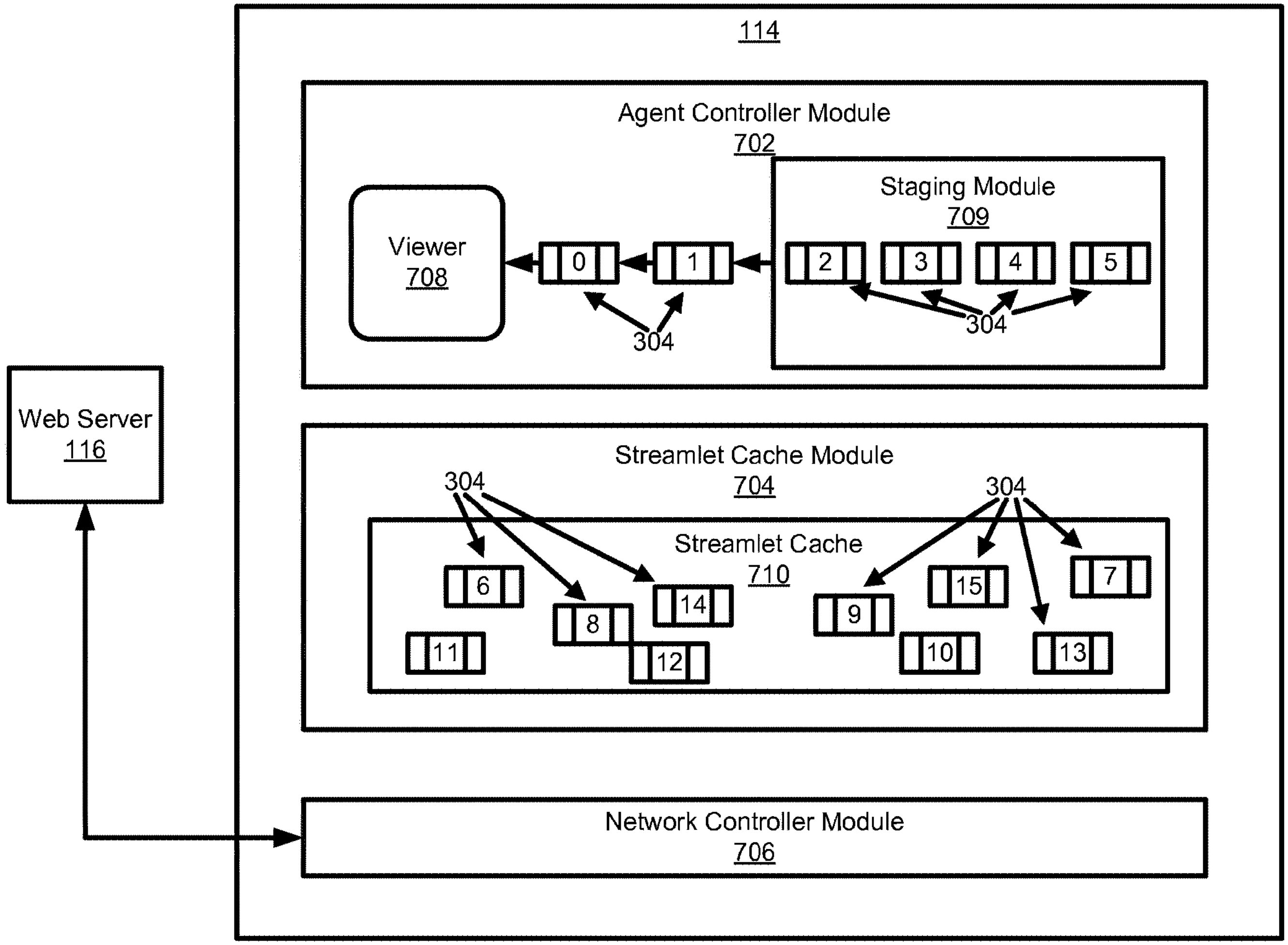

114
Agent Controller Module
702
Viewer
708
0
1
304
Staging Module
709
2
3
4
5
304
Web Server
116
Streamlet Cache Module
704
304
304
Streamlet Cache
710
6
8
14
11
12
9
15
7
10
13
Network Controller Module
706

FIG. 7

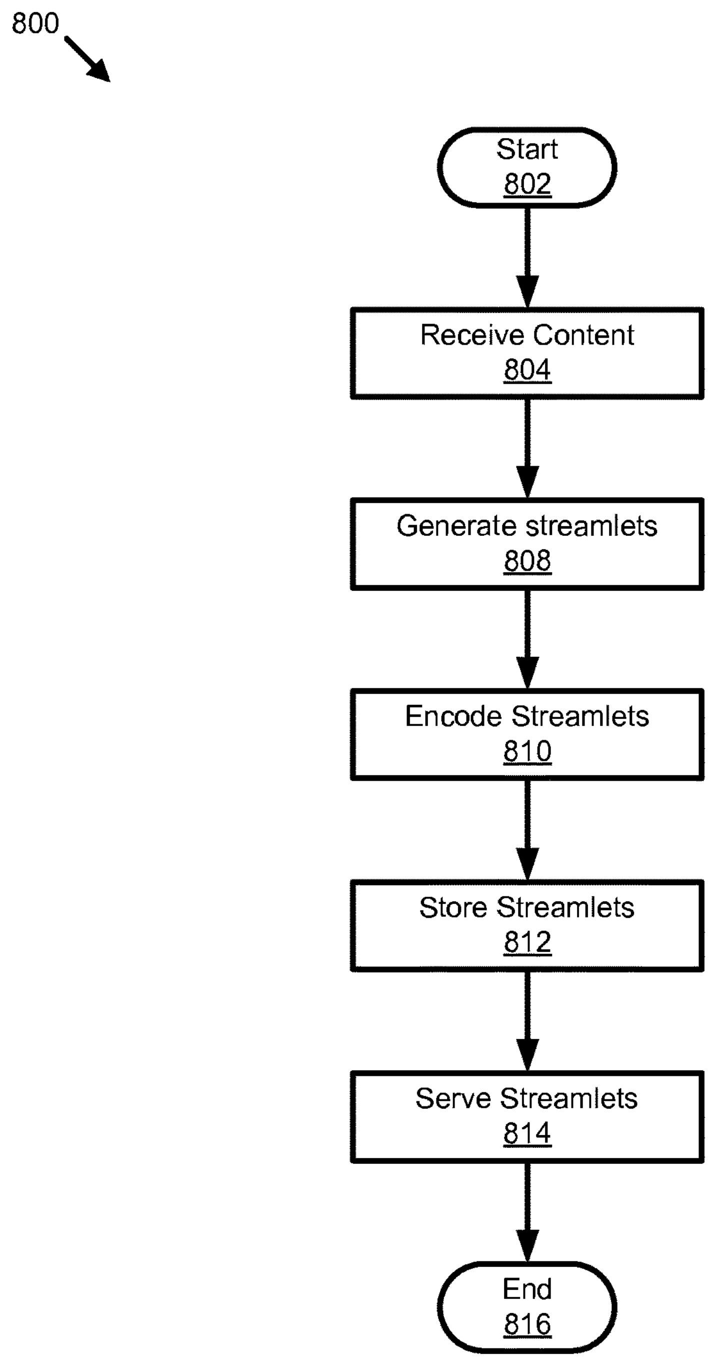

800
Start
802
Receive Content
804
Generate streamlets
808
Encode Streamlets
810
Store Streamlets
812
Serve Streamlets
814
End
816

FIG. 8

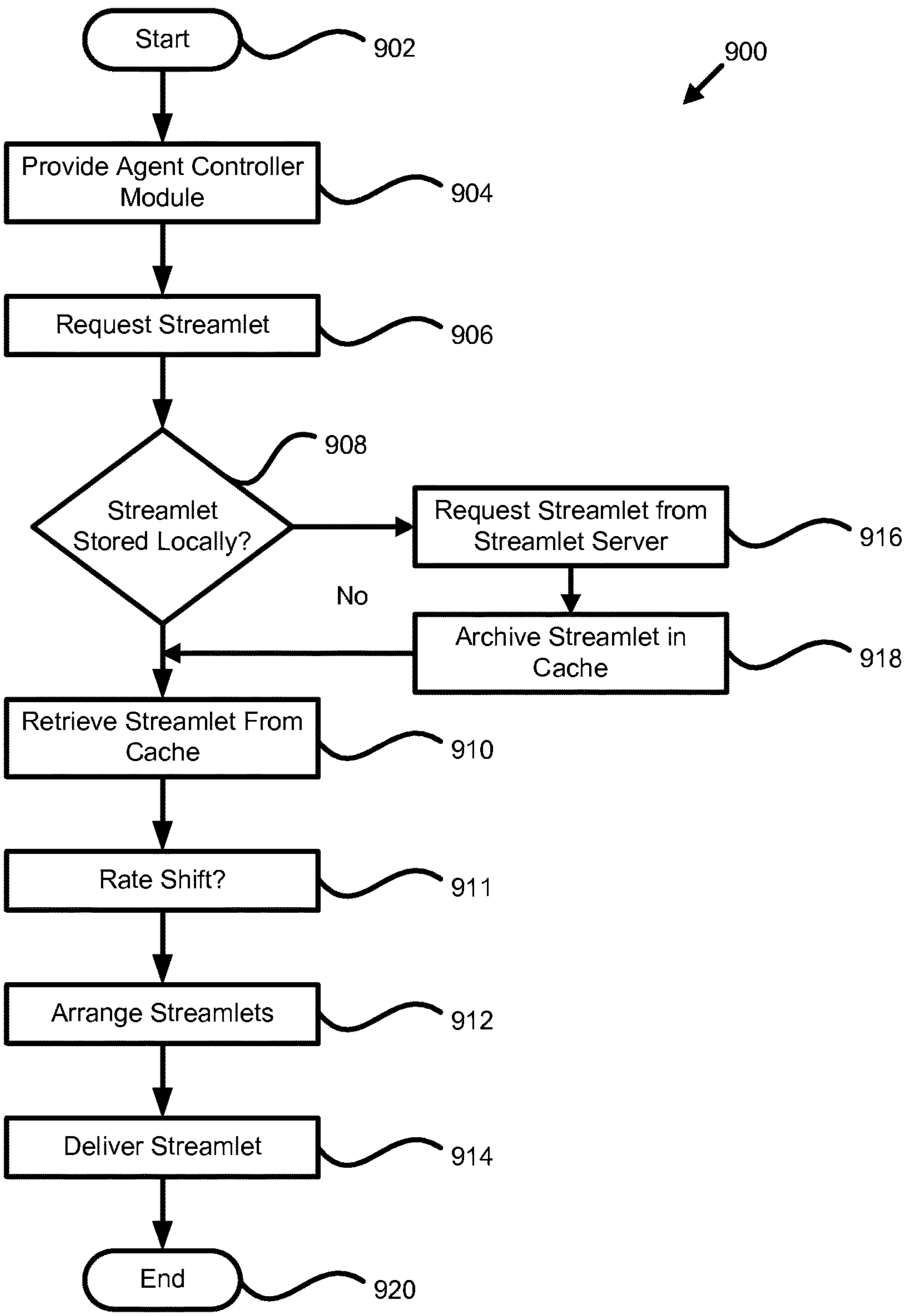

Start
902
900
Provide Agent Controller Module
904
Request Streamlet
906
908
Streamlet Stored Locally?
Request Streamlet from Streamlet Server
916
No
Archive Streamlet in Cache
918
Retrieve Streamlet From Cache
910
Rate Shift?
911
Arrange Streamlets
912
Deliver Streamlet
914
End
920

FIG. 9

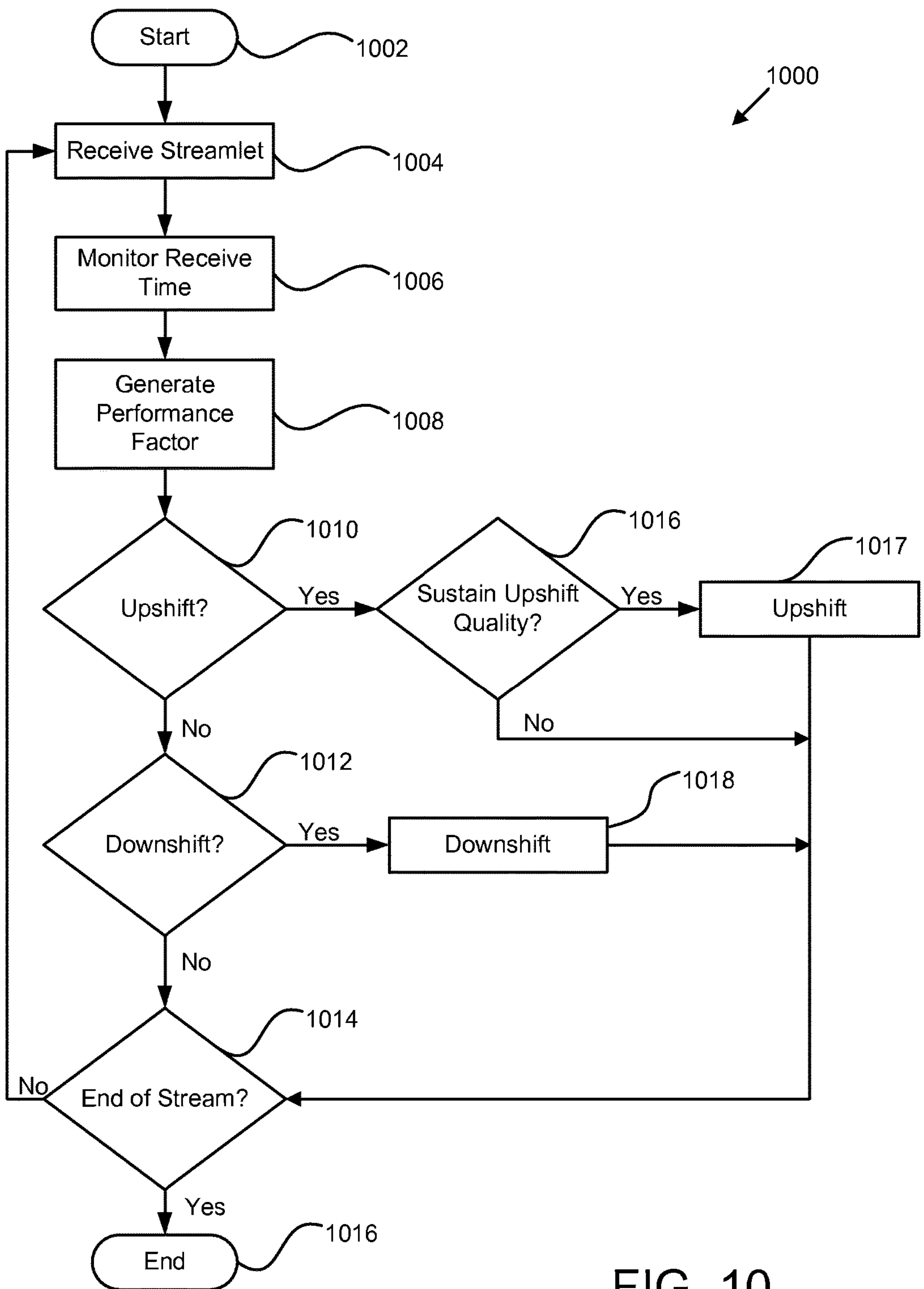

Start
1002
1000
Receive Streamlet
1004
Monitor Receive Time
1006
Generate Performance Factor
1008
Upshift?
1010
Yes
Sustain Upshift Quality?
1016
Yes
Upshift
1017
No
No
Downshift?
1012
Yes
Downshift
1018
No
End of Stream?
1014
No
Yes
End
1016

FIG. 10

APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 16/004,056 filed on Jun. 8, 2018, which is a continuation of U.S. patent application Ser. No. 15/414,027 (now U.S. Pat. No. 9,998,516) filed on Jan. 24, 2017, which is a continuation of U.S. patent application Ser. No. 14/719,122 filed on May 21, 2015, which is a continuation of U.S. patent application Ser. No. 14/106,051 filed on Dec. 13, 2013 (now U.S. Pat. No. 9,071,668), which is a continuation of U.S. patent application Ser. No. 13/617,114, filed on Sep. 14, 2012 (now U.S. Pat. No. 8,612,624), which is a continuation of U.S. patent Ser. No. 12/906,940 filed on Oct. 18, 2010 (now U.S. Pat. No. 8,402,156), which is a continuation of U.S. patent application Ser. No. 11/673,483, filed on Feb. 9, 2007 (now U.S. Pat. No. 7,818,444), which is a continuation-in-part of application Ser. No. 11/116,783, filed on Apr. 28, 2005 (now U.S. Pat. No. 8,868,772), which claims the benefit of U.S. Provisional Application No. 60/566,831, filed on Apr. 31, 2004, all of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to video streaming over packet switched networks such as the Internet, and more particularly relates to adaptive-rate shifting of streaming content over such networks.

Description of the Related Art

The Internet is fast becoming a preferred method for distributing media files to end users. It is currently possible to download music or video to computers, cell phones, or practically any network capable device. Many portable media players are equipped with network connections and enabled to play music or videos. The music or video files (hereinafter "media files") can be stored locally on the media player or computer, or streamed or downloaded from a server.

"Streaming media" refers to technology that delivers content at a rate sufficient for presenting the media to a user in real time as the data is received. The data may be stored in memory temporarily until played and then subsequently deleted. The user has the immediate satisfaction of viewing the requested content without waiting for the media file to completely download. Unfortunately, the audio/video quality that can be received for real time presentation is constrained by the available bandwidth of the user's network connection. Streaming may be used to deliver content on demand (previously recorded) or from live broadcasts.

Alternatively, media files may be downloaded and stored on persistent storage devices, such as hard drives or optical storage, for later presentation. Downloading complete media files can take large amounts of time depending on the network connection. Once downloaded, however, the content can be viewed repeatedly anytime or anywhere. Media files prepared for downloading usually are encoded with a higher quality audio/video than can be delivered in real time. Users generally dislike this option, as they tend to want to see or hear the media file instantaneously.

Streaming offers the advantage of immediate access to the content but currently sacrifices quality compared with downloading a file of the same content. Streaming also provides the opportunity for a user to select different content for viewing on an ad hoc basis, while downloading is by definition restricted to receiving a specific content selection in its entirety or not at all. Downloading also supports rewind, fast forward, and direct seek operations, while streaming is unable to fully support these functions. Streaming is also vulnerable to network failures or congestion.

Another technology, known as "progressive downloads," attempts to combine the strengths of the above two technologies. When a progressive download is initiated, the media file download begins, and the media player waits to begin playback until there is enough of the file downloaded that playback can begin with the hope that the remainder of the file will be completely downloaded before playback "catches up." This waiting period before playback can be substantial depending on network conditions, and therefore is not a complete or fully acceptable solution to the problem of media presentation over a network.

Generally, three basic challenges exist with regard to data transport streaming over a network such as the Internet that has a varying amount of data loss. The first challenge is reliability. Most streaming solutions use a TCP connection, or "virtual circuit," for transmitting data. A TCP connection provides a guaranteed delivery mechanism so that data sent from one endpoint will be delivered to the destination, even if portions are lost and retransmitted. A break in the continuity of a TCP connection can have serious consequences when the data must be delivered in real-time. When a network adapter detects delays or losses in a TCP connection, the adapter "backs off" from transmission attempts for a moment and then slowly resumes the original transmission pace. This behavior is an attempt to alleviate the perceived congestion. Such a slowdown is detrimental to the viewing or listening experience of the user and therefore is not acceptable.

The second challenge to data transport is efficiency. Efficiency refers to how well the user's available bandwidth is used for delivery of the content stream. This measure is directly related to the reliability of the TCP connection. When the TCP connection is suffering reliability problems, a loss of bandwidth utilization results. The measure of efficiency sometimes varies suddenly, and can greatly impact the viewing experience.

The third challenge is latency. Latency is the time measure form the client's point-of-view, of the interval between when a request is issued and the response data begins to arrive. This value is affected by the network connection's reliability and efficiency, and the processing time required by the origin to prepare the response. A busy or overloaded server, for example, will take more time to process a request. As well as affecting the start time of a particular request, latency has a significant impact on the network throughput of TCP.

From the foregoing discussion, it should be apparent that a need exists for an apparatus, system, and method that alleviate the problems of reliability, efficiency, and latency. Additionally, such an apparatus, system, and method would offer instantaneous viewing along with the ability to fast forward, rewind, direct seek, and browse multiple streams. Beneficially, such an apparatus, system, and method would utilize multiple connections between a source and destination, requesting varying bitrate streams depending upon network conditions.

SUMMARY OF THE INVENTION

The present invention has been developed in response to the present state of the art, and in particular, in response to the problems and needs in the art that have not yet been fully solved by currently available content streaming systems. Accordingly, the present invention has been developed to provide an apparatus, system, and method for adaptive-rate content streaming that overcome many or all of the above-discussed shortcomings in the art.

The apparatus for adaptive-rate content streaming is provided with a logic unit containing a plurality of modules configured to functionally execute the necessary steps. These modules in the described embodiments include a receiving module configured to receive media content, a streamlet module configured to segment the media content and generate a plurality of sequential streamlets, and an encoding module configured to encode each streamlet as a separate content file.

The encoding module is further configured to generate a set of streamlets for each of the sequential streamlets. Each streamlet may comprise a portion of the media content having a predetermined length of time. The predetermined length of time may be in the range of between about 0.1 and 5 seconds.

In one embodiment, a set of streamlets comprises a plurality of streamlets having identical time indices, and each streamlet of the set of streamlets has a unique bitrate. The receiving module is configured to convert the media content to raw audio or raw video. The encoding module may include a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid. The job completion bid may be based on a plurality of computing variables selected from a group consisting of current encoding job completion percentage, average encoding job completion time, processor speed, and physical memory capacity.

A system of the present invention is also presented for adaptive-rate content streaming. In particular, the system, in one embodiment, includes a receiving module configured to receive media content, a streamlet module configured to segment the media content and generate a plurality of sequential streamlets, each streamlet comprising a portion of the media content having a predetermined length of time, and an encoding module configured to encode each streamlet as a separate content file and generate a set of streamlets.

The system also includes a plurality of streamlets having identical time indices and each streamlet of the set of streamlets having a unique bitrate. The encoding module comprises a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid.

A method of the present invention is also presented for adaptive-rate content streaming. In one embodiment, the method includes receiving media content, segmenting the media content and generating a plurality of sequential streamlets, and encoding each streamlet as a separate content file.

The method also includes segmenting the media content into a plurality of streamlets, each streamlet comprising a portion of the media content having a predetermined length of time. In one embodiment, the method includes generating a set of streamlets comprising a plurality of streamlets having identical time indices, and each streamlet of the set of streamlets having a unique bitrate.

Furthermore, the method may include converting the media content to raw audio or raw video, and segmenting the content media into a plurality of sequential streamlets. The method further comprises assigning an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid, and submitting an encoding job completion bid based on a plurality of computing variables.

Reference throughout this specification to features, advantages, or similar language does not imply that all of the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussion of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

Furthermore, the described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize that the invention may be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

These features and advantages of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the advantages of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings, in which:

FIG. **1** is a schematic block diagram illustrating one embodiment of a system for dynamic rate shifting of streaming content in accordance with the present invention;

FIG. **2*a*** is a schematic block diagram graphically illustrating one embodiment of a media content file;

FIG. **2*b*** is a schematic block diagram illustrating one embodiment of a plurality of streams having varying degrees of quality and bandwidth;

FIG. **3*a*** is a schematic block diagram illustrating one embodiment of a stream divided into a plurality of source streamlets;

FIG. **3*b*** is a schematic block diagram illustrating one embodiment of sets of streamlets in accordance with the present invention;

FIG. **4** is a schematic block diagram illustrating in greater detail one embodiment of the content module in accordance with the present invention;

FIG. **5*a*** is a schematic block diagram illustrating one embodiment of an encoder module in accordance with the present invention;

FIG. **5***b* is a schematic block diagram illustrating one embodiment of parallel encoding of streamlets in accordance with the present invention:

FIG. **6***a* is a schematic block diagram illustrating one embodiment of a virtual timeline in accordance with the present invention;

FIG. **6***b* is a schematic block diagram illustrating an alternative embodiment of a VT in accordance with the present invention:

FIG. **6***c* is a schematic block diagram illustrating one embodiment of a QMX in accordance with the present invention;

FIG. **7** is a schematic block diagram graphically illustrating one embodiment of a client module in accordance with the present invention:

FIG. **8** is a schematic flow chart diagram illustrating one embodiment of a method for processing content in accordance with the present invention;

FIG. **9** is a schematic flow chart diagram illustrating one embodiment of a method for viewing a plurality of streamlets in accordance with the present invention; and

FIG. **10** is a schematic flow chart diagram illustrating one embodiment of a method for requesting streamlets within an adaptive-rate shifting content streaming environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Many of the functional units described in this specification have been labeled as modules, in order to more particularly emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom VLSI circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices or the like.

Modules may also be implemented in software for execution by various types of processors. An identified module of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions which may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may comprise disparate instructions stored in different locations which, when joined logically together, comprise the module and achieve the stated purpose for the module.

Indeed, a module of executable code may be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

Reference throughout this specification to "one embodiment," "an embodiment." or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in one embodiment," "in an embodiment," and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

Reference to a signal bearing medium may take any form capable of generating a signal, causing a signal to be generated, or causing execution of a program of machine-readable instructions on a digital processing apparatus. A signal bearing medium may be embodied by a transmission line, a compact disk, digital-video disk, a magnetic tape, a Bernoulli drive, a magnetic disk, a punch card, flash memory, integrated circuits, or other digital processing apparatus memory device. In one embodiment, a computer program product including a computer useable medium having a computer readable program of computer instructions stored thereon that when executed on a computer causes the computer to carry out operations for multi-bitrate content streaming as described herein.

Furthermore, the described features, structures, or characteristics of the invention may be combined in any suitable manner in one or more embodiments. In the following description, numerous specific details are provided, such as examples of programming, software modules, user selections, network transactions, database queries, database structures, hardware modules, hardware circuits, hardware chips, etc., to provide a thorough understanding of embodiments of the invention. One skilled in the relevant art will recognize, however, that the invention may be practiced without one or more of the specific details, or with other methods, components, materials, and so forth. In other instances, well-known structures, materials, or operations are not shown or described in detail to avoid obscuring aspects of the invention.

FIG. **1** is a schematic block diagram illustrating one embodiment of a system **100** for dynamic rate shifting of streaming content in accordance with the present invention. In one embodiment, the system **100** comprises a content server **102** and an end user station **104**. The content server **102** and the end user station **104** may be coupled by a data communications network. The data communications network may include the Internet **106** and connections **108** to the Internet **106**. Alternatively, the content server **102** and the end user **104** may be located on a common local area network, wireless area network, cellular network, virtual local area network, or the like. The end user station **104** may comprise a personal computer (PC), an entertainment system configured to communicate over a network, or a portable electronic device configured to present content. For example, portable electronic devices may include, but are not limited to, cellular phones, portable gaming systems, and portable computing devices.

In the depicted embodiment, the system **100** also includes a publisher **110**, and a web server **116**. The publisher **110** may be a creator or distributor of content. For example, if the content to be streamed were a broadcast of a television program, the publisher **110** may be a television or cable network channel such as NBC®, or MTV®. Content may be transferred over the Internet **106** to the content server **102**, where the content is received by a content module **112**. The content module **112** may be configured to receive, process, and store content. In one embodiment, processed content is accessed by a client module **114** configured to play the content on the end user station **104**. In a further embodiment, the client module **114** is configured to receive different portions of a content stream from a plurality of locations simultaneously. For example, the client module **114** may request and receive content from any of the plurality of web servers **116**.

Content from the content server **102** may be replicated to other web servers **116** or alternatively to proxy cache servers **118**. Replicating may occur by deliberate forwarding from the content server **102**, or by a web, cache, or proxy server outside of the content server **102** asking for content on behalf of the client module **114**. In a further embodiment, content may be forwarded directly to web **116** or proxy **118** servers through direct communication channels **120** without the need to traverse the Internet **106**.

FIG. **2***a* is a schematic block diagram graphically illustrating one embodiment of a media content (hereinafter "content") file **200**. In one embodiment, the content file **200** is distributed by the publisher **110**. The content file **200** may comprise a television broadcast, sports event, movie, music, concert, etc. The content file **200** may also be live or archived content. The content file **200** may comprise uncompressed video and audio, or alternatively, video or audio. Alternatively, the content file **200** may be compressed using standard or proprietary encoding schemes. Examples of encoding schemes capable of use with the present invention include, but are not limited to, DivX®, Windows Media Video®, Quicktime Sorenson 3®, On2, OGG Vorbis. MP3, or Quicktime 6.5/MPEG-4® encoded content.

FIG. **2***b* is a schematic block diagram illustrating one embodiment of a plurality of streams **202** having varying degrees of quality and bandwidth. In one embodiment, the plurality of streams **202** comprises a low quality stream **204**, a medium quality stream **206**, and a high quality stream **208**. Each of the streams **204**, **206**, **208** is a copy of the content file **200** encoded and compressed to varying bit rates. For example, the low quality stream **204** may be encoded and compressed to a bit rate of 100 kilobits per second (kbps), the medium quality stream **206** may be encoded and compressed to a bit rate of 200 kbps, and the high quality stream **208** may be encoded and compressed to 600 kbps.

FIG. **3***a* is a schematic block diagram illustrating one embodiment of a stream **302** divided into a plurality of source streamlets **303**. As used herein, streamlet refers to any sized portion of the content file **200**. Each streamlet **303** may comprise a portion of the content contained in stream **302**, encapsulated as an independent media object. The content in a streamlet **303** may have a unique time index in relation to the beginning of the content contained in stream **302**. In one embodiment, the content contained in each streamlet **303** may have a duration of two seconds. For example, streamlet 0 may have a time index of 00:00 representing the beginning of content playback, and streamlet 1 may have a time index of 00:02, and so on. Alternatively, the time duration of the streamlets **304** may be any duration smaller than the entire playback duration of the content in stream **302**. In a further embodiment, the streamlets **303** may be divided according to file size instead of a time index and duration.

FIG. **3***b* is a schematic block diagram illustrating one embodiment of sets **306** of streamlets in accordance with the present invention. As used herein, the term "set" refers to a group of streamlets having identical time indices and durations but varying bitrates. In the depicted embodiment, the set **306***a* encompasses all streamlets having a time index of 00:00. The set **306***a* includes encoded streamlets **304** having low, medium, and high **204**, **206**, **208** bitrates. Of course each set **306** may include more than the depicted three bitrates which are given by way of example only. One skilled in the art will recognize that any number of streams having different bitrates may be generated from the original content **200**.

As described above, the duration of one streamlet **304** may be approximately two seconds. Likewise each set **306** may comprise a plurality of streamlets **304** where each streamlet **304** has a playable duration of two seconds. Alternatively, the duration of the streamlet **304** may be predetermined or dynamically variable depending upon a variety of factors including, but not limited to, network congestion, system specifications, playback resolution and quality, etc. In the depicted embodiment, the content **200** may be formed of the plurality of sets **306**. The number of sets **306** may depend on the length of the content **200** and the length or duration of each streamlet **304**.

FIG. **4** is a schematic block diagram illustrating in greater detail one embodiment of the content module **112** in accordance with the present invention. The content module **112** may comprise a capture module **402**, a streamlet module **404**, an encoder module **406**, a streamlet database **408**, and the web server **116**. In one embodiment, the capture module **402** is configured to receive the content file **200** from the publisher **110**. The capture module **402** may be configured to "decompress" the content file **200**. For example, if the content file **200** arrives having been encoded with one of the above described encoding schemes, the capture module **402** may convert the content file **200** into raw audio and/or video. Alternatively, the content file **200** may be transmitted by the publisher in a format **110** that does not require decompression.

The capture module **402** may comprise a capture card configured for TV and/or video capture. One example of a capture card suitable for use in the present invention is the DRC-2500 by Digital Rapids of Ontario, Canada. Alternatively, any capture card capable of capturing audio and video may be utilized with the present invention. In a further embodiment, the capture module **402** is configured to pass the content file to the streamlet module **404**.

The streamlet module **404**, in one embodiment, is configured to segment the content file **200** and generate source streamlets **303** that are not encoded. As used herein, the term "segment" refers to an operation to generate a streamlet of the content file **200** having a duration or size equal to or less than the duration or size of the content file **200**. The streamlet module **404** may be configured to segment the content file **200** into streamlets **303** each having an equal duration. Alternatively, the streamlet module **404** may be configured to segment the content file **200** into streamlets **303** having equal file sizes.

The encoding module **406** is configured to receive the source streamlets **303** and generate the plurality of streams **202** of varying qualities. The original content file **200** from the publisher may be digital in form and may comprise content having a high bit rate such as, for example, 2 mbps. The content may be transferred from the publisher **110** to the content module **112** over the Internet **106**. Such transfers of data are well known in the art and do not require further discussion herein. Alternatively, the content may comprise a captured broadcast.

In a further embodiment, the encoding module **406** is configured to generate a plurality of sets **306** of streamlets **304**. The sets **306**, as described above with reference to FIG. **3***b*, may comprise streamlets having an identical time index and duration, and a unique bitrate. As with FIG. **3***b*, the sets **306** and subsequently the plurality of streams **202** may comprise the low quality stream **204**, the medium quality stream **206**, and the high quality stream **208**. Alternatively, the plurality of streams **202** may comprise any number of streams deemed necessary to accommodate end user bandwidth.

The encoder module **406** is further configured to encode each source streamlet **303** into the plurality of streams **202** and streamlet sets **306** and store the streamlets in the streamlet database **408**. The encoding module **406** may utilize encoding schemes such as DivX®, Windows Media Video 9®, Quicktime 6.5 Sorenson 3®, or Quicktime 6.5/MPEG-4®. Alternatively, a custom encoding scheme may be employed.

The content module **112** may also include a metadata module **412** and a metadata database **414**. In one embodiment, metadata comprises static searchable content information. For example, metadata includes, but is not limited to, air date of the content, title, actresses, actors, length, and episode name. Metadata is generated by the publisher **110**, and may be configured to define an end user environment. In one embodiment, the publisher **100** may define an end user navigational environment for the content including menus, thumbnails, sidebars, advertising, etc. Additionally, the publisher **110** may define functions such as fast forward, rewind, pause, and play that may be used with the content file **200**. The metadata module **412** is configured to receive the metadata from the publisher **110** and store the metadata in the metadata database **414**. In a further embodiment, the metadata module **412** is configured to interface with the client module **114**, allowing the client module **114** to search for content based upon at least one of a plurality of metadata criteria. Additionally, metadata may be generated by the content module **112** through automated process(es) or manual definition.

Once the streamlets **304** have been received and processed, the client module **114** may request streamlets **304** using HTTP from the web server **116**. Using a standard protocol such as HTTP eliminates the need for network administrators to configure firewalls to recognize and pass through network traffic for a new, specialized protocol. Additionally, since the client module **114** initiates the request, the web server **116** is only required to retrieve and serve the requested streamlet **304**. In a further embodiment, the client module **114** may be configured to retrieve streamlets **304** from a plurality of web servers **116**.

Each web server **116** may be located in various locations across the Internet **106**. The streamlets **304** may essentially be static files. As such, no specialized media server or server-side intelligence is required for a client module **114** to retrieve streamlets **304**. Streamlets **304** may be served by the web server **116** or cached by cache servers of Internet Service Providers (ISPs), or any other network infrastructure operators, and served by the cache server. Use of cache servers is well known to those skilled in the art, and will not be discussed further herein. Thus, a highly scalable solution is provided that is not hindered by massive amounts of client module **114** requests to the web server **116** at any specific location, especially the web server **116** most closely associated with or within the content module **112**

FIG. **5***a* is a schematic block diagram illustrating one embodiment of an encoder module **406** in accordance with the present invention. In one embodiment, the encoder module **406** may include a master module **502** and a plurality of host computing modules (hereinafter "host") **504**. The hosts **504** may comprise personal computers, servers, etc. In a further embodiment, the hosts **504** may be dedicated hardware, for example, cards plugged into a single computer.

The master module (hereinafter "master") **502** is configured to receive streamlets **303** from the streamlet module **404** and stage the streamlet **303** for processing. In one embodiment, the master **502** may decompress each source streamlet **303** to produce a raw streamlet. As used herein, the term "raw streamlet" refers to a streamlet **303** that is uncompressed or lightly compressed to substantially reduce size with no significant loss in quality. A lightly compressed raw streamlet can be transmitted more quickly and to more hosts. Each host **504** is coupled with the master **502** and configured to receive a raw streamlet from the master **502** for encoding. The hosts **504**, in one example, generate a plurality of streamlets **304** having identical time indices and durations, and varying bitrates. Essentially each host **504** may be configured to generate a set **306** from the raw streamlet **503** sent from the master **502**. Alternatively, each host **504** may be dedicated to producing a single bitrate in order to reduce the time required for encoding.

Upon encoding completion, the host **504** returns the set **306** to the master **502** so that the encoding module **406** may store the set **306** in the streamlet database **408**. The master **502** is further configured to assign encoding jobs to the hosts **504**. Each host is configured to submit an encoding job completion bid (hereinafter "bid"). The master **502** assigns encoding jobs depending on the bids from the hosts **504**. Each host **504** generates a bid depending upon a plurality of computing variables which may include, but are not limited to, current encoding job completion percentage, average job completion time, processor speed and physical memory capacity.

For example, a host **504** may submit a bid that indicates that based on past performance history the host **504** would be able to complete the encoding job in 15 seconds. The master **502** is configured to select from among a plurality of bids the best bid and subsequently submit the encoding job to the host **504** with the best bid. As such, the described encoding system does not require that each host **504** have identical hardware but beneficially takes advantage of the available computing power of the hosts **504**. Alternatively, the master **502** selects the host **504** based on a first come first serve basis, or some other algorithm deemed suitable for a particular encoding job.

The time required to encode one streamlet **304** is dependent upon the computing power of the host **504**, and the encoding requirements of the content file **200**. Examples of encoding requirements may include, but are not limited to, two or multi-pass encoding, and multiple streams of different bitrates. One benefit of the present invention is the ability to perform two-pass encoding on a live content file **200**. Typically, in order to perform two-pass encoding prior art systems must wait for the content file to be completed before encoding.

The present invention, however, segments the content file **200** into source streamlets **303** and the two-pass encoding to a plurality of streams **202** may be performed on each corresponding raw streamlet without waiting for a TV show to end, for example. As such, the content module **112** is capable of streaming the streamlets over the Internet shortly after the content module **112** begins capture of the content file **200**. The delay between a live broadcast transmitted from the publisher **110** and the availability of the content depends on the computing power of the hosts **504**.

FIG. **5***b* is a schematic block diagram illustrating one embodiment of parallel encoding of streamlets in accordance with the present invention. In one example, the capture module **402** (of FIG. **4**) begins to capture the content file and the streamlet module **404** generates a first streamlet **303***a* and passes the streamlet to the encoding module **406**. The encoding module **406** may take 10 seconds, for example, to generate the first set **306***a* of streamlets **304***a* (**304***a***1**, **304***a***2**, **304***a***3**, etc. represent streamlets **304** of

different bitrates). FIG. **5***b* illustrates the encoding process generically as block **502** to graphically illustrate the time duration required to process a raw or lightly encoded streamlet **303** as described above with reference to the encoding module **406**. The encoding module **406** may simultaneously process more than one streamlet **303**, and processing of streamlets will begin upon arrival of the streamlet from the capture module **402**.

During the 10 seconds required to encode the first streamlet **303***a*, the streamlet module **404** has generated five additional 2-second streamlets **303***b*, **303***c*, **303***d*, **303***e*, **303***f*, for encoding and the master **502** has prepared and staged the corresponding raw streamlets. Two seconds after the first set **306***a* is available the next set **306***b* is available, and so on. As such, the content file **200** is encoded for streaming over the Internet and appears live. The 10 second delay is given herein by way of example only. Multiple hosts **504** may be added to the encoding module **406** in order to increase the processing capacity of the encoding module **406**. The delay may be shortened to an almost unperceivable level by the addition of high CPU powered systems, or alternatively multiple low powered systems.

A system as described above beneficially enables multi-pass encoding of live events. Multi-pass encoding systems of the prior art require that the entire content be captured (or be complete) because in order to perform multi-pass encoding the entire content must be scanned and processed more than once. This is impossible with prior art systems because content from a live event is not complete until the event is over. As such, with prior art systems, multi-pass encoding can only be performed once the event is over. Streamlets, however, may be encoded as many times as is deemed necessary. Because the streamlet is an encapsulated media object of 2 seconds (for example), multi-pass encoding may begin on a live event once the first streamlet is captured. Shortly after multi-pass encoding of the first streamlet **303***a* is finished, multi-pass encoding of the second streamlet **303***b* finishes, and as such multi-pass encoding is performed on a live event and appears live to a viewer.

Any specific encoding scheme applied to a streamlet may take longer to complete than the time duration of the streamlet itself, for example, a very high quality encoding of a 2-second streamlet may take 5 seconds to finish. Alternatively, the processing time required for each streamlet may be less than the time duration of a streamlet. However, because the offset parallel encoding of successive streamlets are encoded by the encoding module at regular intervals (matching the intervals at which the those streamlets are submitted to the encoding module **406**, for example 2 seconds) the output timing of the encoding module **406** does not fall behind the real-time submission rate of the unencoded streamlets. Conversely, prior art encoding systems rely on the very fastest computing hardware and software because the systems must generate the output immediately in lock-step with the input. A prior art system that takes 2.1 seconds to encode 2 seconds worth of content is considered a failure. The present invention allows for slower than real-time encoding processes yet still achieves a real-time encoding effect due to the parallel offset pipes.

The parallel offset pipeline approach described with reference to FIG. **5***b* beneficially allows for long or short encoding times without "falling behind" the live event. Additionally, arbitrarily complex encoding of streamlets to multiple profiles and optimizations only lengthens the encoding time **502** without a perceptible difference to a user because the sets **306** of streamlets **304** are encoded in a time-selective manner so that streamlets are processed at regular time intervals and transmitted at these time intervals.

Returning now to FIG. **5***a*, as depicted, the master **502** and the hosts **504** may be located within a single local area network, or in other terms, the hosts **504** may be in close physical proximity to the master **502**. Alternatively, the hosts **504** may receive encoding jobs from the master **502** over the Internet or other communications network. For example, consider a live sports event in a remote location where it would be difficult to setup multiple hosts. In this example, a master performs no encoding or alternatively light encoding before publishing the streamlets online. The hosts **504** would then retrieve those streamlets and encode the streamlets into the multiple bitrate sets **306** as described above.

Furthermore, hosts **504** may be dynamically added or removed from the encoding module without restarting the encoding job and/or interrupting the publishing of streamlets. If a host **504** experiences a crash or some failure, its encoding work is simply reassigned to another host.

The encoding module **406**, in one embodiment, may also be configured to produce streamlets that are specific to a particular playback platform. For example, for a single raw streamlet, a single host **504** may produce streamlets for different quality levels for personal computer playback, streamlets for playback on cell phones with a different, proprietary codec, a small video-only streamlet for use when playing just a thumbnail view of the stream (like in a programming guide), and a very high quality streamlet for use in archiving.

FIG. **6***a* is a schematic block diagram illustrating one embodiment of a virtual timeline **600** in accordance with the present invention. In one embodiment, the virtual timeline **600** comprises at least one quantum media extension **602**. The quantum media extension (hereinafter "QMX") **602** describes an entire content file **200**. Therefore, the virtual timeline (hereinafter "VT") **600** may comprise a file that is configured to define a playlist for a user to view. For example, the VT may indicate that the publisher desires a user to watch a first show QMX **602***a* followed by QMX **602***b* and QMX **602***c*. As such, the publisher may define a broadcast schedule in a manner similar to a television station.

FIG. **6***b* is a schematic block diagram illustrating an alternative embodiment of a VT **600** in accordance with the present invention. In the depicted embodiment, the VT **600** may include a single QMX **602** which indicates that the publisher desires the same content to be looped over and over again. For example, the publisher may wish to broadcast a never-ending infomercial on a website.

FIG. **6***c* is a schematic block diagram illustrating one embodiment of a QMX **602** in accordance with the present invention. In one embodiment, the QMX **602** contains a multitude of information generated by the content module **112** configured to describe the content file **200**. Examples of information include, but are not limited to, start index **604**, end index **606**, whether the content is live **608**, proprietary publisher data **610**, encryption level **612**, content duration **614** and bitrate values **616**. The bitrate values **616** may include frame size **618**, audio channel **620** information, codecs **622** used, sample rate **624**, and frames parser **626**.

A publisher may utilize the QVT **600** together with the QMX **602** in order to prescribe a playback order for users, or alternatively selectively edit content. For example, a publisher may indicate in the QMX **602** that audio should be muted at time index 10:42 or video should be skipped for 3 seconds at time index 18:35. As such, the publisher may

selectively skip offensive content without the processing requirements of editing the content.

FIG. **7** is a schematic block diagram graphically illustrating one embodiment of a client module **114** in accordance with the present invention. The client module **114** may comprise an agent controller module **702**, a streamlet cache module **704**, and a network controller module **706**. In one embodiment, the agent controller module **702** is configured to interface with a viewer **708**, and transmit streamlets **304** to the viewer **708**. Alternatively, the agent controller module **702** may be configured to simply reassemble streamlets into a single file for transfer to an external device such as a portable video player.

In a further embodiment, the client module **114** may comprise a plurality of agent controller modules **702**. Each agent controller module **702** may be configured to interface with one viewer **708**. Alternatively, the agent controller module **702** may be configured to interface with a plurality of viewers **708**. The viewer **708** may be a media player (not shown) operating on a PC or handheld electronic device.

The agent controller module **702** is configured to select a quality level of streamlets to transmit to the viewer **708**. The agent controller module **702** requests lower or higher quality streams based upon continuous observation of time intervals between successive receive times of each requested streamlet. The method of requesting higher or lower quality streams will be discussed in greater detail below with reference to FIG. **10**.

The agent controller module **702** may be configured to receive user commands from the viewer **708**. Such commands may include play, fast forward, rewind, pause, and stop. In one embodiment, the agent controller module **702** requests streamlets **304** from the streamlet cache module **704** and arranges the received streamlets **304** in a staging module **709**. The staging module **709** may be configured to arrange the streamlets **304** in order of ascending playback time. In the depicted embodiment, the streamlets **304** are numbered 0, 1, 2, 3, 4, etc. However, each streamlet **304** may be identified with a unique filename.

Additionally, the agent controller module **702** may be configured to anticipate streamlet **304** requests and pre-request streamlets **304**. By pre-requesting streamlets **304**, the user may fast-forward, skip randomly, or rewind through the content and experience no buffering delay. In a further embodiment, the agent controller module **702** may request the streamlets **304** that correspond to time index intervals of 30 seconds within the total play time of the content. Alternatively, the agent controller module **702** may request streamlets at any interval less than the length of the time index. This enables a "fast-start" capability with no buffering wait when starting or fast-forwarding through content file **200**. In a further embodiment, the agent controller module **702** may be configured to pre-request streamlets **304** corresponding to specified index points within the content or within other content in anticipation of the end user **104** selecting new content to view. In one embodiment, the streamlet cache module **704** is configured to receive streamlet **304** requests from the agent controller module **702**. Upon receiving a request, the streamlet cache module **704** first checks a streamlet cache **710** to verify if the streamlet **304** is present. In a further embodiment, the streamlet cache module **704** handles streamlet **304** requests from a plurality of agent controller modules **702**. Alternatively, a streamlet cache module **704** may be provided for each agent controller module **702**. If the requested streamlet **304** is not present in the streamlet cache **410**, the request is passed to the network controller module **706**. In order to enable fast forward and rewind capabilities, the streamlet cache module **704** is configured to store the plurality of streamlets **304** in the streamlet cache **710** for a specified time period after the streamlet **304** has been viewed. However, once the streamlets **304** have been deleted, they may be requested again from the web server **116**.

The network controller module **706** may be configured to receive streamlet requests from the streamlet cache module **704** and open a connection to the web server **116** or other remote streamlet **304** database (not shown). In one embodiment, the network controller module **706** opens a TCP/IP connection to the web server **116** and generates a standard HTTP GET request for the requested streamlet **304**. Upon receiving the requested streamlet **304**, the network controller module **706** passes the streamlet **304** to the streamlet cache module **704** where it is stored in the streamlet cache **710**. In a further embodiment, the network controller module **706** is configured to process and request a plurality of streamlets **304** simultaneously. The network controller module **706** may also be configured to request a plurality of streamlets, where each streamlet **304** is subsequently requested in multiple parts.

In a further embodiment, streamlet requests may comprise requesting pieces of any streamlet file. Splitting the streamlet **304** into smaller pieces or portions beneficially allows for an increased efficiency potential, and also eliminates problems associated with multiple full-streamlet requests sharing the bandwidth at any given moment. This is achieved by using parallel TCP/IP connections for pieces of the streamlets **304**. Consequently, efficiency and network loss problems are overcome, and the streamlets arrive with more useful and predictable timing.

In one embodiment, the client module **114** is configured to use multiple TCP connections between the client module **114** and the web server **116** or web cache. The intervention of a cache may be transparent to the client or configured by the client as a forward cache. By requesting more than one streamlet **304** at a time in a manner referred to as "parallel retrieval," or more than one part of a streamlet **304** at a time, efficiency is raised significantly and latency is virtually eliminated. In a further embodiment, the client module allows a maximum of three outstanding streamlet **304** requests. The client module **114** may maintain additional open TCP connections as spares to be available should another connection fail. Streamlet **304** requests are rotated among all open connections to keep the TCP flow logic for any particular connection from falling into a slow-start or close mode. If the network controller module **706** has requested a streamlet **304** in multiple parts, with each part requested on mutually independent TCP/IP connections, the network controller module **706** reassembles the parts to present a complete streamlet **304** for use by all other components of the client module **114**.

When a TCP connection fails completely, a new request may be sent on a different connection for the same streamlet **304**. In a further embodiment, if a request is not being satisfied in a timely manner, a redundant request may be sent on a different connection for the same streamlet **304**. If the first streamlet request's response arrives before the redundant request response, the redundant request can be aborted. If the redundant request response arrives before the first request response, the first request may be aborted.

Several streamlet **304** requests may be sent on a single TCP connection, and the responses are caused to flow back in matching order along the same connection. This eliminates all but the first request latency. Because multiple responses are always being transmitted, the processing

latency of each new streamlet **304** response after the first is not a factor in performance. This technique is known in the industry as "pipelining." Pipelining offers efficiency in request-response processing by eliminating most of the effects of request latency. However, pipelining has serious vulnerabilities. Transmission delays affect all of the responses. If the single TCP connection fails, all of the outstanding requests and responses are lost. Pipelining causes a serial dependency between the requests.

Multiple TCP connections may be opened between the client module **114** and the web server **116** to achieve the latency-reduction efficiency benefits of pipelining while maintaining the independence of each streamlet **304** request. Several streamlet **304** requests may be sent concurrently, with each request being sent on a mutually distinct TCP connection. This technique is labeled "virtual pipelining" and is an innovation of the present invention. Multiple responses may be in transit concurrently, assuring that communication bandwidth between the client module **114** and the web server **116** is always being utilized. Virtual pipelining eliminates the vulnerabilities of traditional pipelining. A delay in or complete failure of one response does not affect the transmission of other responses because each response occupies an independent TCP connection. Any transmission bandwidth not in use by one of multiple responses (whether due to delays or TCP connection failure) may be utilized by other outstanding responses.

A single streamlet **304** request may be issued for an entire streamlet **304**, or multiple requests may be issued, each for a different part or portion of the streamlet. If the streamlet is requested in several parts, the parts may be recombined by the client module **114** streamlet.

In order to maintain a proper balance between maximized bandwidth utilization and response time, the issuance of new streamlet requests must be timed such that the web server **116** does not transmit the response before the client module **114** has fully received a response to one of the previously outstanding streamlet requests. For example, if three streamlet **304** requests are outstanding, the client module **114** should issue the next request slightly before one of the three responses is fully received and "out of the pipe." In other words, request timing is adjusted to keep three responses in transit. Sharing of bandwidth among four responses diminishes the net response time of the other three responses. The timing adjustment may be calculated dynamically by observation, and the request timing adjusted accordingly to maintain the proper balance of efficiency and response times.

The schematic flow chart diagrams that follow are generally set forth as logical flow chart diagrams. As such, the depicted order and labeled steps are indicative of one embodiment of the presented method. Other steps and methods may be conceived that are equivalent in function, logic, or effect to one or more steps, or portions thereof, of the illustrated method. Additionally, the format and symbols employed are provided to explain the logical steps of the method and are understood not to limit the scope of the method. Although various arrow types and line types may be employed in the flow chart diagrams, they are understood not to limit the scope of the corresponding method. Indeed, some arrows or other connectors may be used to indicate only the logical flow of the method. For instance, an arrow may indicate a waiting or monitoring period of unspecified duration between enumerated steps of the depicted method. Additionally, the order in which a particular method occurs may or may not strictly adhere to the order of the corresponding steps shown.

FIG. **8** is a schematic flow chart diagram illustrating one embodiment of a method **800** for processing content in accordance with the present invention. In one embodiment the method **800** starts **802**, and the content module **112** receives **804** content from the publisher **110**. Receiving content **804** may comprise receiving **804** a digital copy of the content file **200**, or digitizing a physical copy of the content file **200**. Alternatively, receiving **804** content may comprise capturing a radio, television, cable, or satellite broadcast. Once received **804**, the streamlet module **404** generates **808** a plurality of source streamlets **303** each having a fixed duration. Alternatively, the streamlets **303** may be generated with a fixed file size.

In one embodiment, generating **808** streamlets comprises dividing the content file **200** into a plurality of two second streamlets **303**. Alternatively, the streamlets may have any length less than or equal to the length of the stream **202**. The encoder module **406** then encodes **810** the streamlets **303** into sets **306** of streamlets **304**, in a plurality of streams **202** according to an encoding scheme. The quality may be predefined, or automatically set according to end user bandwidth, or in response to pre-designated publisher guidelines

In a further embodiment, the encoding scheme comprises a proprietary codec such as WMV9®. The encoder module **406** then stores **812** the encoded streamlets **304** in the streamlet database **408**. Once stored **812**, the web server **116** may then serve **814** the streamlets **304**. In one embodiment, serving **814** the streamlets **304** comprises receiving streamlet requests from the client module **114**, retrieving the requested streamlet **304** from the streamlet database **408**, and subsequently transmitting the streamlet **304** to the client module **114**. The method **800** then ends **816**.

FIG. **9** is a schematic flow chart diagram illustrating one embodiment of a method **900** for viewing a plurality of streamlets in accordance with the present invention. The method **900** starts and an agent controller module **702** is provided **904** and associated with a viewer **708** and provided with a staging module **709**. The agent controller module **702** then requests **906** a streamlet **304** from the streamlet cache module **704**. Alternatively, the agent controller module **702** may simultaneously request **906** a plurality of streamlets **304** the streamlet cache module **704**. If the streamlet is stored **908** locally in the streamlet cache **710**, the streamlet cache module **704** retrieves **910** the streamlet **304** and sends the streamlet to the agent controller module **702**. Upon retrieving **910** or receiving a streamlet, the agent controller module **702** makes **911** a determination of whether or not to shift to a higher or lower quality stream **202**. This determination will be described below in greater detail with reference to FIG. **10**.

In one embodiment, the staging module **709** then arranges **912** the streamlets **304** into the proper order, and the agent controller module **702** delivers **914** the streamlets to the viewer **708**. In a further embodiment, delivering **914** streamlets **304** to the end user comprises playing video and or audio streamlets on the viewer **708**. If the streamlets **304** are not stored **908** locally, the streamlet request is passed to the network controller module **706**. The network controller module **706** then requests **916** the streamlet **304** from the web server **116**. Once the streamlet **304** is received, the network controller module **706** passes the streamlet to the streamlet cache module **704**. The streamlet cache module **704** archives **918** the streamlet. Alternatively, the streamlet cache module **704** then archives **918** the streamlet and passes the streamlet to the agent controller module **702**, and the method **900** then continues from operation **910** as described above.

Referring now to FIG. **10**, shown therein is a schematic flow chart diagram illustrating one embodiment of a method **1000** for requesting streamlets **304** within an adaptive-rate shifting content streaming environment in accordance with the present invention. The method **1000** may be used in one embodiment as the operation **911** of FIG. **9**. The method **1000** starts and the agent controller module **702** receives **1004** a streamlet **304** as described above with reference to FIG. **9**. The agent controller module **702** then monitors **1006** the receive time of the requested streamlet. In one embodiment, the agent controller module **702** monitors the time intervals Δ between successive receive times for each streamlet response. Ordering of the responses in relation to the order of their corresponding requests is not relevant.

Because network behavioral characteristics fluctuate, sometimes quite suddenly, any given Δ may vary substantially from another. In order to compensate for this fluctuation, the agent controller module **702** calculates **1008** a performance ratio r across a window of n samples for streamlets of playback length S. In one embodiment, the performance ratio r is calculated using the equation:

$$r = S\frac{n}{\sum_{i=1}^{n} \Delta_i}$$

Due to multiple simultaneous streamlet processing, and in order to better judge the central tendency of the performance ratio r, the agent controller module **702** may calculate a geometric mean, or alternatively an equivalent averaging algorithm, across a window of size m, and obtain a performance factor φ:

$$\varphi_{current} = \left(\prod_{j=1}^{m} r_j\right)^{\frac{1}{m}}$$

The policy determination about whether or not to upshift **1010** playback quality begins by comparing $\varphi_{current}$ with a trigger threshold Θ_{up}. If $\varphi_{current} \geq \Theta_{up}$, then an up shift to the next higher quality stream may be considered **1016**. In one embodiment, the trigger threshold Θ_{up} is determined by a combination of factors relating to the current read ahead margin (i.e. the amount of contiguously available streamlets that have been sequentially arranged by the staging module **709** for presentation at the current playback time index), and a minimum safety margin. In one embodiment, the minimum safety margin may be 24 seconds. The smaller the read ahead margin, the larger Θ_{up} is to discourage upshifting until a larger read ahead margin may be established to withstand network disruptions. If the agent controller module **702** is able to sustain **1016** upshift quality, then the agent controller module **702** will upshift **1017** the quality and subsequently request higher quality streams. The determination of whether use of the higher quality stream is sustainable **1016** is made by comparing an estimate of the higher quality stream's performance factor, φ_{higher}, with Θ_{up}. If $\varphi_{higher} \geq \Theta_{up}$ then use of the higher quality stream is considered sustainable. If the decision of whether or not the higher stream rate is sustainable **1016** is "no," the agent controller module **702** will not attempt to upshift **1017** stream quality. If the end of the stream has been reached **1014**, the method **1000** ends **1016**.

If the decision on whether or not to attempt upshift **1010** is "no", a decision about whether or not to downshift **1012** is made. In one embodiment, a trigger threshold Θ_{down} is defined in a manner analogous to Θ_{up}. If $\varphi_{current} > \Theta_{down}$ then the stream quality may be adequate, and the agent controller module **702** does not downshift **1018** stream quality. However, if $\varphi_{current} \leq \Theta_{down}$, the agent controller module **702** does downshift **1018** the stream quality. If the end of the stream has not been reached **1014**, the agent controller module **702** begins to request and receive **1004** lower quality streamlets and the method **1000** starts again. Of course, the above described equations and algorithms are illustrative only, and may be replaced by alternative streamlet monitoring solutions.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. A system for adaptive-rate content streaming of live event video playable on one or more end user stations over the Internet, the system comprising:

at least one storage device storing live event video, the live event video encoded at a plurality of different bitrates creating a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, the low quality stream, the medium quality stream, and the high quality stream each comprising a group of streamlets encoded at a respective one of the plurality of different bitrates, each group of streamlets comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the live event video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and

wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the live event video in each of the low quality stream, the medium quality stream, and the high quality stream, and wherein the first streamlet of the low quality stream encodes the same first portion of the live event video at a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream.

2. The system of claim **1**, wherein the second streamlet of each of the groups of streamlets each has the same second duration and corresponds to the same second portion of the live event video in the low quality stream, the medium quality stream, and the high quality stream, the second streamlet of the low quality stream having the same bitrate as the first streamlet of the low quality stream.

3. The system of claim **2**, wherein the first and second durations are different.

4. The system of claim **1**, further comprising:

a plurality of web servers located at different locations across the internet, each web server configured to:

receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing a portion of the video, wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one

of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams;

retrieve from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

send the retrieved first streamlet from the currently selected one of the different copies to the requesting one of the end user stations over the one or more network connections.

5. The system of claim **1**, wherein each of the first streamlets has a first duration that is the range of 0.1 to 5 seconds.

6. The system of claim **1**, wherein the live event is a live sports event.

7. The system of claim **1**, further comprising:

a first web server configured to:

receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the live event video, wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the live event video;

retrieve from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

send the retrieved first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream to the requesting one of the end user stations over the one or more network connections.

8. The system of claim **7**, wherein the first streamlets of the low quality stream, the medium quality stream, and the high quality stream are available before the live event is complete.

9. The system of claim **7**, wherein the streamlets of the low quality stream, the medium quality stream, and the high quality stream of the live event are available on a 10 second delay.

10. The system of claim **7**, wherein the streamlets from the low quality stream, the medium quality stream, and the high quality stream of the live event, when played back, appear live to a viewer.

11. The system of claim **7**, wherein the first web server is further configured to:

receive at least one virtual timeline request over the one or more internet connections from the one or more end user stations to retrieve a virtual timeline; and

send the virtual timeline to the requesting one of the end user stations over the one or more network connections.

12. The system of claim **11**, wherein the virtual timeline corresponds to the currently selected one of the low quality stream, the medium quality stream, and the high quality stream.

13. The system of claim **11**, wherein the virtual timeline defines a playlist for a user to view.

14. The system of claim **11**, wherein the virtual timeline comprises a file that is configured to define a playlist for a user to view.

15. The system of claim **11**, wherein the virtual timeline comprises at least one quantum media extension (QMX).

16. An end user station to stream a live event video over a network from a server for playback of the video, the content player device comprising:

a processor;

a digital processing apparatus memory device comprising non-transitory machine-readable instructions that, when executed, cause the processor to:

establish one or more network connections between the end user station and the server, wherein the server is configured to access at least one of a plurality of groups of streamlets;

wherein the live event video is encoded at a plurality of different bitrates to create a plurality of streams including at least a low quality stream, a medium quality stream, and a high quality stream, each of the low quality stream, the medium quality stream, and the high quality stream comprising a group of streamlets encoded at the same respective one of the different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the live event video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bit rate of no less than 600 kbps; and

wherein the first streamlets of each of the low quality stream, the medium quality stream and the high quality stream each has an equal playback duration and each of the first streamlets encodes the same portion of the live event video at a different one of the different bitrates;

select a specific one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams;

place a streamlet request to the server over the one or more network connections for the first streamlet of the selected stream;

receive the requested first streamlet from the server via the one or more network connections; and

provide the received first streamlet for playback of the live event video.

17. The end user station of claim **16**, wherein the second streamlet of each of the groups of streamlets each has the same second duration and corresponds to the same second portion of the live event video in the low quality stream, the medium quality stream, and the high quality stream, the second streamlet of the low quality stream having the same bitrate as the first streamlet of the low quality stream.

18. The end user station of claim **17**, wherein the first and second durations are different.

19. The end user station of claim **16**, wherein each of the first streamlets has a first duration that is the range of 0.1 to 5 seconds.

20. The end user station of claim **16**, wherein the first streamlets of the low quality stream, the medium quality stream, and the high quality stream are available before the live event is complete.

21. The end user station of claim **16**, wherein the streamlets of the low quality stream, the medium quality stream, and the high quality stream of the live event are available on a ten second delay.

22. The end user station of claim **16**, wherein the streamlets from the low quality stream, the medium quality stream, and the high quality stream of the live event, when played back, appear live to a viewer.

23. The end user station of claim **16**, wherein the end user station is further configured to:

request and receive a virtual timeline; and

wherein one or more streamlet requests are based on the at least one virtual timeline.

24. The end user station of claim **23**, wherein the virtual timeline corresponds to the currently selected one of the low quality stream, the medium quality stream, and the high quality stream.

25. The end user station of claim **23**, wherein the virtual timeline defines a playlist for a user to view.

26. A process executable by one or more servers to stream a live event video for playback by one or more end user stations, the process comprising:

storing, by the one or more servers, a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the live event video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and

wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the live event video in the low quality stream, the medium quality stream, and the high quality stream, the first streamlet of the low quality stream having a different one of the different bitrates than the first streamlet of the high quality stream and the first streamlet of the medium quality stream;

receiving at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the live event video, wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the live event video;

retrieving from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

sending the retrieved first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream to the requesting one of the end user stations over the one or more network connections.

27. The process of claim **26**, wherein the second streamlet of each of the groups of streamlets each has the same second duration and corresponds to the same second portion of the live event video in the low quality stream, the medium quality stream, and the high quality stream, the second streamlet of the low quality stream having the same bitrate as the first streamlet of the low quality stream.

28. The process of claim **26**, wherein the first and second durations are different.

29. The process of claim **26**, wherein the first streamlets of the low quality stream, the medium quality stream, and the high quality stream are available before the live event is complete.

30. A process executable by a content player device to stream a live event video over a network from a server for playback of the video by the content player device, the process comprising:

establishing one or more network connections between the content player device and the server, wherein the server accesses a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the live event video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and

wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the live event video in the low quality stream, the medium quality stream, and the high quality stream, the first streamlet of the low quality stream having a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream;

selecting, by the content player device, a currently selected one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the live event video;

placing a streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the live event video;

receiving the requested streamlet from the server via the one or more network connections; and

rendering, by the content player device, the received streamlet for playback of the live event video.

* * * * *

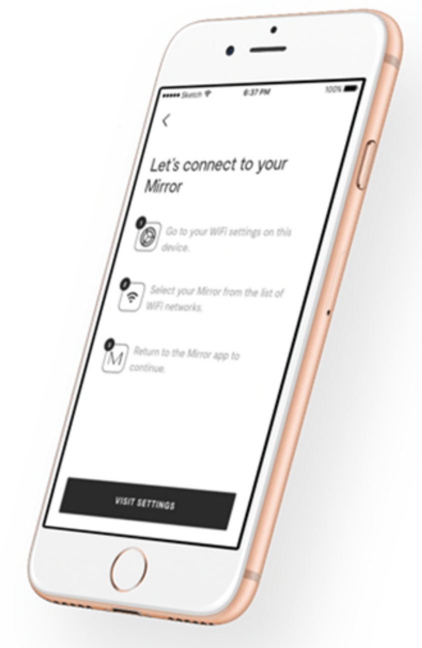
EXHIBIT B-1

U.S. Patent No. 10,469,554 to Mirror

The following claim chart shows exemplary aspects of the Mirror Application and Mirror Device that infringe the claim below. The chart is exemplary and should not be read to limit DISH's claims against Mirror to the specific products or services described below. The chart should also not be read to limit DISH's claims to the patent claim charted below. Nor should the chart below be read to limit how the Mirror Application and Mirror Devices infringe the claim below.

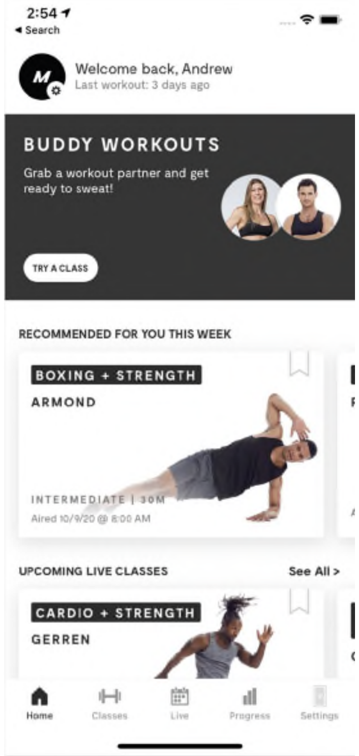
Claim	Claim Limitation	Example Infringement Evidence
16	An end user station to stream a live event video over a network from a server for playback of the video, the content player device comprising:	The Mirror Application is software that permits "an end user station to stream a live event video over a network from a server for playback of the video." The Mirror Application is executable by devices that are end user stations and it obtains streams of a selected live event video for playback of the video. The streams are obtained by the Mirror Application from the Mirror Server(s) over a network. The Mirror Devices are "end user station[s] to stream a live event video over a network from a server for playback of the video." The Mirror Devices obtain streams of a selected video program for playback of the video on the Mirror Device's content player, as shown above. The streams are obtained over a network. The exemplary images in this chart are from the Mirror Device or the Mirror Application running on an Apple iPhone XS (Mirror's iOS Application). **MIRROR DIGITAL OVERVIEW** Access all of MIRROR's Live and On Demand Classes 24/7 from your phone, tablet, and smart TV devices. This feature is included for free with the MIRROR Membership and access is available immediately (whether you already have your Mirror or are awaiting delivery). Please note, it is not currently possible to purchase MIRROR Digital if you do not own a Mirror. https://mirror.kustomer.help/en_us/mirror-digital-overview-B1pnVd8UU

USP 10,469,554 to Mirror

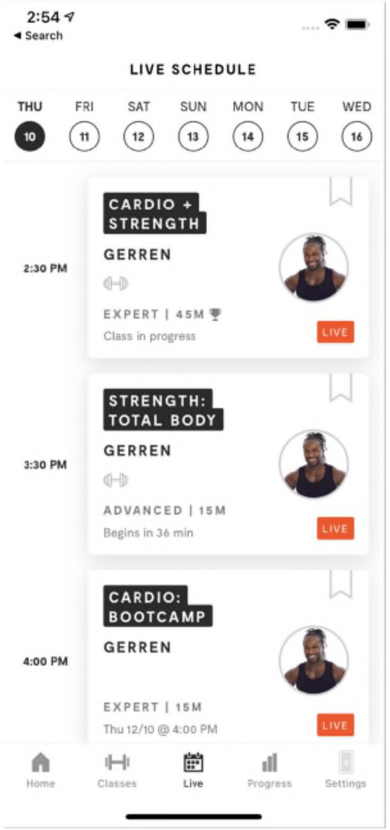
Claim	Claim Limitation	Example Infringement Evidence
		GET THE MIRROR APP To get started setting up your Mirror, you need to download the MIRROR app from the Apple App Store or Google Play Store. The app will take you through everything you need to know. Available on the App Store Get it on Google play Need help? Email us at hello@mirror.co  https://www.mirror.co/app. **MIRROR APP** The MIRROR App allows you to access and customize the Mirror experience. The MIRROR App is available for both iOS and Android! • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-app--S1dDC_tYm.

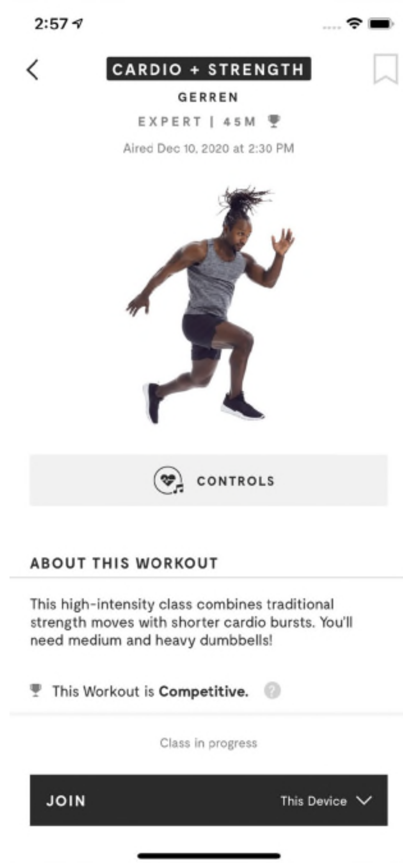
USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Apple App Store When launched, the Mirror Application displays a main menu:

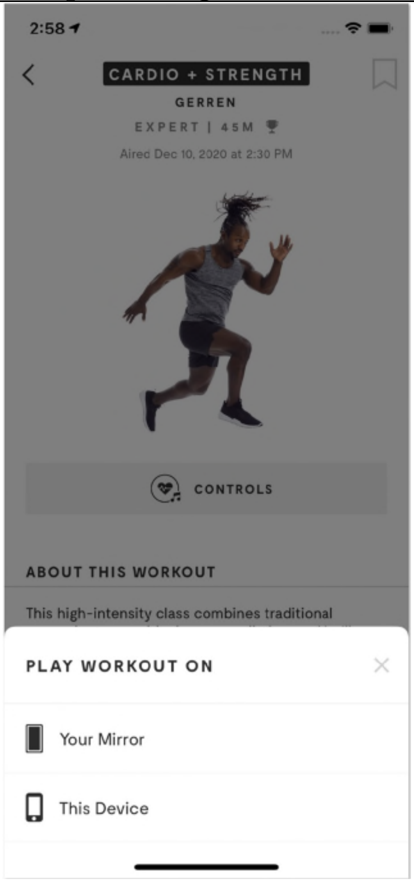
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application The main menu of the Mirror Application displays classes that are each an “event video.” The “Upcoming Live Classes” section of the Mirror Application main menu displays a preview of ongoing and upcoming live event video programming.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting an ongoing live class from the list causes the Mirror Application to display more details regarding the class and provides the option to join the class.

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting the "Join" button for an ongoing live event causes the Mirror iOS Application to provide options to stream the class to a variety of end user stations over the Internet, including the iOS device that the user is using or the separate Mirror Device.

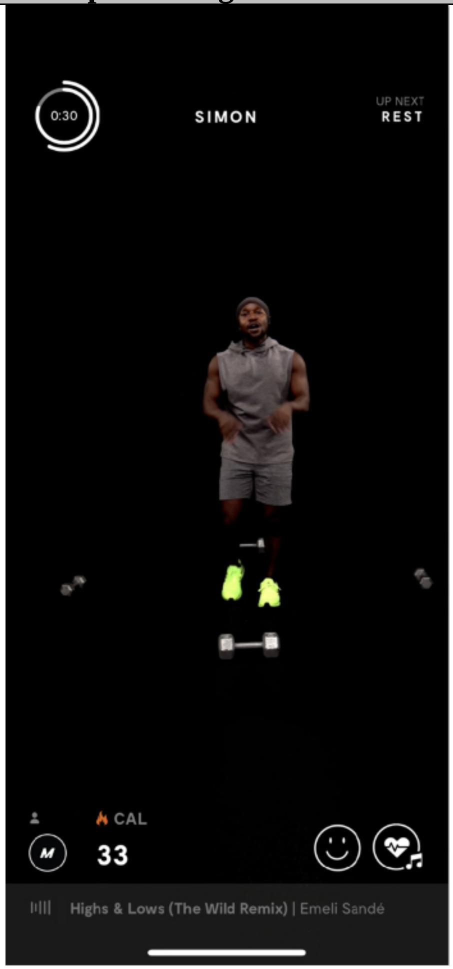
USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting "Your Mirror" causes the selected live event video and other materials to be streamed on the user's Mirror device, which is connected to the Internet.

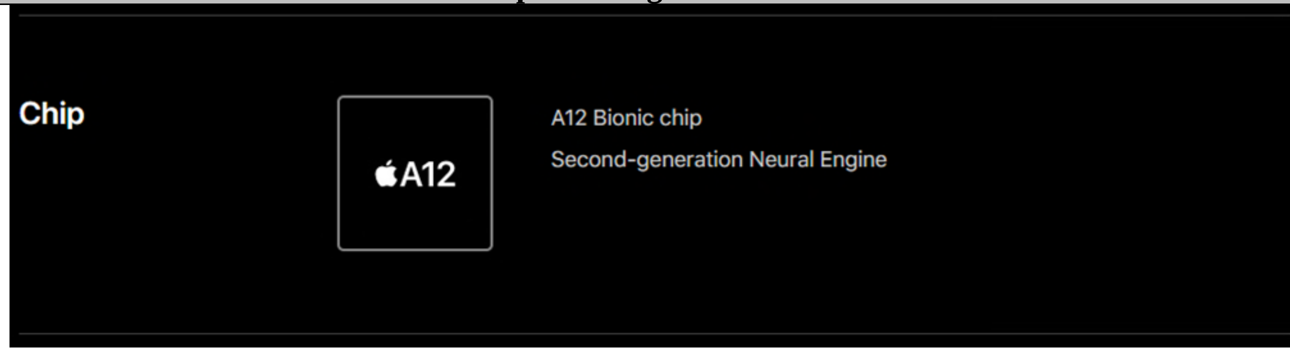
USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Alternatively, selecting “This Device” causes the selected workout video and other materials to be streamed on the user’s iOS device:

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application As shown above, the Mirror Devices are "end user station[s] to stream a live event video over a network from a server for playback of the video." The Mirror Devices obtain streams of a selected video program for playback of the video on the Mirror Device's content player, as shown above. The streams are obtained over a network.

Claim	Claim Limitation	Example Infringement Evidence
	a processor;	A device running the Mirror Application includes at least one "processor" to execute the Mirror Application and stream the live event video. The devices that are compatible with the Mirror Application each include one or more processors. **MIRROR DIGITAL COMPATIBLE DEVICES** The MIRROR App is available for the iPhone, iPad, Android phones, and Android tablets. MIRROR Digital can be casted to your smart TV using these devices. • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-digital-compatible-devices-HklDdOU8U. For example, Mirror requires users to provide a user device such as an iPhone that includes a processor to execute the Mirror Application. **Information** Seller: Refine Fitness LLC Size: 99.3 MB Category: Health & Fitness Compatibility: Requires iOS 11.0 and watchOS 3.2 or later. Compatible with iPhone, iPad, and iPod touch. Languages: English Age Rating: 4+ Copyright: © 2018 Curiouser Products Inc Price: Free In-App Purchases: 1. 1 Year Subscription for Mirror $599.99 Developer Website ↗ App Support ↗ Privacy Policy ↗ https://apps.apple.com/us/app/mirror-workout-companion/id1153358600.

Claim	Claim Limitation	Example Infringement Evidence
		 https://www.apple.com/iphone-xr/specs/. The Mirror Devices also include a processor.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		HARDWARE FRAME — Carbon steel frame Mineral bronze powder coated DISPLAY — 40" full HD 1080p display, with 178° wide viewing angle TECHNOLOGY — Quad core processor SOUND — 2 x 10 watt high-fidelity stereo speakers Embedded omnidirectional microphone CAMERA — 5 megapixel front-facing camera POWER — 1 ft and 6 ft right angle UL certified cables https://www.mirror.co/shop/mirror.
	a digital processing apparatus memory device comprising non-transitory machine-readable instructions that,	The device executes the Mirror Application from "a digital processing apparatus memory device comprising non-transitory machine-readable instructions." The instructions include at least the executable instructions for the Mirror Application and its features. Mirror requires users of the Mirror Application to provide a device with a digital processing apparatus memory device to store the instructions.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td>when executed, cause the processor to:</td><td>For example, Mirror requires users to provide at least 99.3 MB of storage on a digital processing apparatus memory device of the end user station for storing the Mirror Application.

Information
<table>
<tr><td>Seller</td><td>Refine Fitness LLC</td><td></td></tr>
<tr><td>Size</td><td>99.3 MB</td><td></td></tr>
<tr><td>Category</td><td>Health & Fitness</td><td></td></tr>
<tr><td>Compatibility</td><td>Requires iOS 11.0 and watchOS 3.2 or later. Compatible with iPhone, iPad, and iPod touch.</td><td></td></tr>
<tr><td>Languages</td><td>English</td><td></td></tr>
<tr><td>Age Rating</td><td>4+</td><td></td></tr>
<tr><td>Copyright</td><td>© 2018 Curiouser Products Inc</td><td></td></tr>
<tr><td>Price</td><td>Free</td><td></td></tr>
<tr><td>In-App Purchases</td><td>1. 1 Year Subscription for Mirror</td><td>$599.99</td></tr>
<tr><td></td><td>Developer Website ↗ App Support ↗ Privacy Policy ↗</td><td></td></tr>
</table>
https://apps.apple.com/us/app/mirror-workout-companion/id1153358600.

The Mirror Devices also include “a digital processing apparatus memory device comprising non-transitory machine-readable instructions.” For example, the on-board quad core processor requires memory containing non-transitory machine-readable instructions in order to process and display streaming fitness classes.</td></tr>
</table>

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		HARDWARE FRAME — Carbon steel frame Mineral bronze powder coated DISPLAY — 40" full HD 1080p display, with 178° wide viewing angle TECHNOLOGY — Quad core processor SOUND — 2 x 10 watt high-fidelity stereo speakers Embedded omnidirectional microphone CAMERA — 5 megapixel front-facing camera POWER — 1 ft and 6 ft right angle UL certified cables https://www.mirror.co/shop/mirror.
	establish one or more network connections between the end user station and the server, wherein the server is configured to access	As shown below, the non-transitory machine-readable instructions of the Mirror Application and Mirror Devices, when executed, cause the processor(s) to "establish one or more network connections between the end user station and the server" that is "configured to access at least one of a plurality of groups of streamlets." The "segments" discussed herein are "streamlets." The Mirror Application requires an internet connection.

Claim	Claim Limitation	Example Infringement Evidence
	at least one of a plurality of groups of streamlets;	**PRELOAD CLASSES ON MIRROR DIGITAL** You currently cannot preload classes on the MIRROR Digital, however this feature is coming soon! If you are not able to connect WiFi or are in a tough WiFi environment, you can always use cellular data to stream classes on the MIRROR App. Please consult your cell phone provider for questions about data usage and your plan. https://mirror.kustomer.help/en_us/preload-classes-on-mirror-app-H12XPdUUL. To stream a live event video, such as that shown above, the Mirror Application requests a stream of a selected live event video via a network connection . A user may select to stream a live event video by selecting the Join button, as shown above. When the Mirror Application accesses a selected live program, it requests and receives a playlist file that shows the available versions of the program at different resolutions. For the following test, a live event video was selected. In the test, the Mirror Application makes an HTTPS GET request for a master playlist named "**playlist.m3u8**" that specifies the available streams and provides links to the playlists for those streams. The following master playlist named "**playlist.m3u8**" is returned.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080
../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720
../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480
../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360
../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288
../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180
../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
```

Filename: playlist.m3u8.

This is a master playlist file according to the HLS specification.[1] The playlist shows six versions of the stream, denoted by each #EXT-X-STREAM-INF tag at the following bandwidths:

- 6434112 (referred to herein as "**6434112 Bandwidth**")
- 864048 (referred to herein as "**864048 Bandwidth**")
- 403824 (referred to herein as "**403824 Bandwidth**")
- 367728 (referred to herein as "**367728 Bandwidth**")
- 312832 (referred to herein as "**312832 Bandwidth**")
- 249664 (referred to herein as "**249664 Bandwidth**")

For each of these versions, the master playlist provides a link to a playlist file for the specified version of the selected live event video at a particular bandwidth and resolution, which is called a "variant" in HLS.
</td></tr>
</table>

[1] RFC 8216 (HLS Live Streaming), Section 4.3.4 (Master Playlist Tags)

USP 10,469,554 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>The Mirror Application initially selects the 6434112 Bandwidth (1080p – high bandwidth) version of the stream and makes a request for the corresponding variant playlist file named "chunklist.m3u8." That file is returned with the following contents (a portion of which is shown below).

#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1232
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:08.356Z
#EXTINF:2.0,
r4vhrugx/00000000/media_1232.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1233.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1234.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1235.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1236.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1237.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1238.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1239.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1240.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1241.ts
#EXTINF:2.0,

Path: https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		As noted above, the variant playlist file is an HLS playlist. The variant playlist file identifies a plurality of segments or "streamlets" that are part of the **6434112 Bandwidth** group of streamlets. Each line in the file "**chunklist.m3u8**" that begins with "#EXTINF" specifies the length of the segments in seconds (2.0). The line below the #EXTINF entry is the relative location of the video file for the segment (e.g., r4vhrugx/0000000/media_1232.ts). The Mirror Application uses HTTPS GET requests to retrieve the segments of the encoded live event video specified in the file above. The Mirror Application makes a request for a segment, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected live event. As long as the viewer stays on the selected live event video and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **6434112 Bandwidth** version). While the example above relates to the **6434112 Bandwidth** group of streamlets, other groups of streamlets are also available. If the available bandwidth for the network connection decreases, for example, the Mirror Application will continue to request segments to continue streaming the live event program, but at one of the lower bandwidths. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variant playlist file for the **403824 Bandwidth** named "chunklist.m3u8." That file is returned with the following contents showing a portion of the **403824 Bandwidth** group of segments for the live event video being streamed.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1238 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z #EXTINF:2.0, fbd862nq/00000000/media_1238.ts #EXTINF:2.0, fbd862nq/00000000/media_1239.ts #EXTINF:2.0, fbd862nq/00000000/media_1240.ts #EXTINF:2.0, fbd862nq/00000000/media_1241.ts #EXTINF:2.0, fbd862nq/00000000/media_1242.ts #EXTINF:2.0, fbd862nq/00000000/media_1243.ts #EXTINF:2.0, fbd862nq/00000000/media_1244.ts #EXTINF:2.0, fbd862nq/00000000/media_1245.ts #EXTINF:2.0, fbd862nq/00000000/media_1246.ts #EXTINF:2.0, fbd862nq/00000000/media_1247.ts #EXTINF:2.0, fbd862nq/00000000/media_1248.ts Filename: **chunklist.m3u8** The Mirror Application then makes the request for media_1238.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected live event video at the **403824 Bandwidth**. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror Application will continue to request and

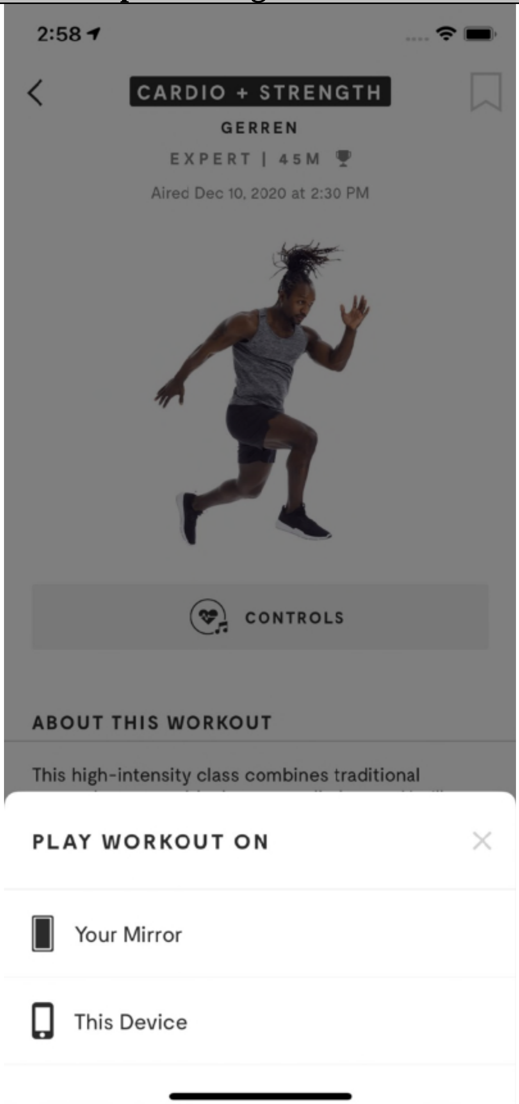
Claim	Claim Limitation	Example Infringement Evidence
		receive playlists corresponding to the current, chosen resolution (in this case, the **403824 Bandwidth** version). As the bandwidth is further constrained, the Mirror Application makes another request for a lower quality stream. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variable playlist file for the **249664 Bandwidth** named "chunklist.m3u8." That file is returned with the following contents showing a portion of the **249664 Bandwidth** group of segments for the live event video being streamed. #EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1241 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:26.354Z #EXTINF:2.0, zf4q4ivl/00000000/media_1241.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1242.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1243.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1244.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1245.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1246.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1247.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1248.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1249.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1250.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1251.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1252.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1253.ts

Claim	Claim Limitation	Example Infringement Evidence
		File: **chunklist.m3u8** The Mirror Application then makes the request for media_1281.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected live event video at the **249664 Bandwidth**. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **249664 Bandwidth** version). A portion of a subsequently retrieved playlist's contents are shown below.

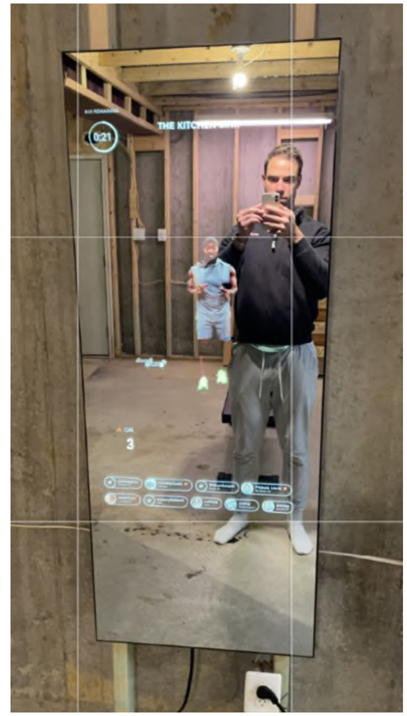
Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1245 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:34.354Z #EXTINF:2.0, zf4q4ivl/00000000/media_1245.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1246.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1247.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1248.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1249.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1250.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1251.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1252.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1253.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1254.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1255.ts #EXTINF:2.0, Filename: **chunklist.m3u8** The subsequently retrieved playlist includes additional video segments that were not included in the previous playlist file, for example: "**media_1254.ts**" and "**media_1255.ts.**" The Mirror Application continues to request, receive, and playback successive segments of the live event video to stream the live event video. The Mirror Devices also require an internet connection.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		CONNECTION INTERNET: Dual-band 802.11 A/B/G/N Wi-Fi APP: Controlled by iOS or Android companion app HEART RATE: Syncs with Bluetooth™ heart rate monitors, Apple Watches, and Android Wear OS Watches AUDIO: Pairs with Bluetooth™ speakers and headphones https://www.mirror.co/shop/mirror To stream a live event video, such as that shown above, the Mirror Device requests a stream of a selected live event video via a network connection . The iOS application provides the interface for interacting with a Mirror Device (i.e., selecting a live or on demand class to stream). After selecting a class, the user selects whether to use the iOS device or Mirror Device to view the content (i.e., stream and participate in the workout).

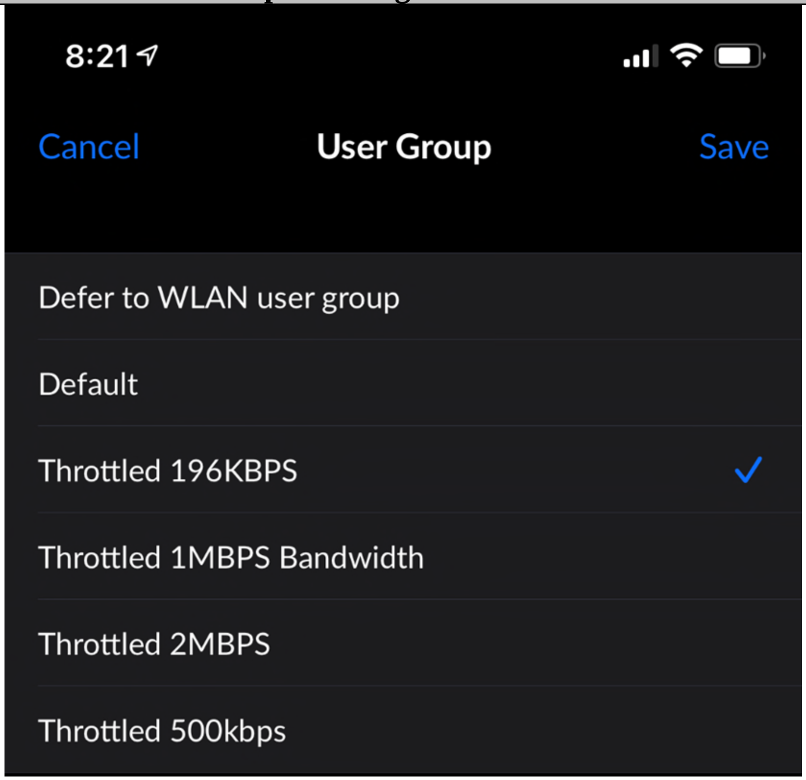
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Selecting "Your Mirror" causes the Mirror Device to initiate streaming requests:  For the following test, a live programming event was selected. Based on the test, and upon information and belief, the Mirror Devices operates in the same or substantially the same way as the Mirror Application. For example, when the Mirror Device(s) accesses a selected live event video, the Mirror Device(s) initially selects a first bandwidth version of the stream, makes a request for the segments of the group corresponding to the selected bandwidth version of the live event program, receives segments from the group corresponding to the selected bandwidth version, and then plays the requested and received segments on the Mirror Device content player as shown below.

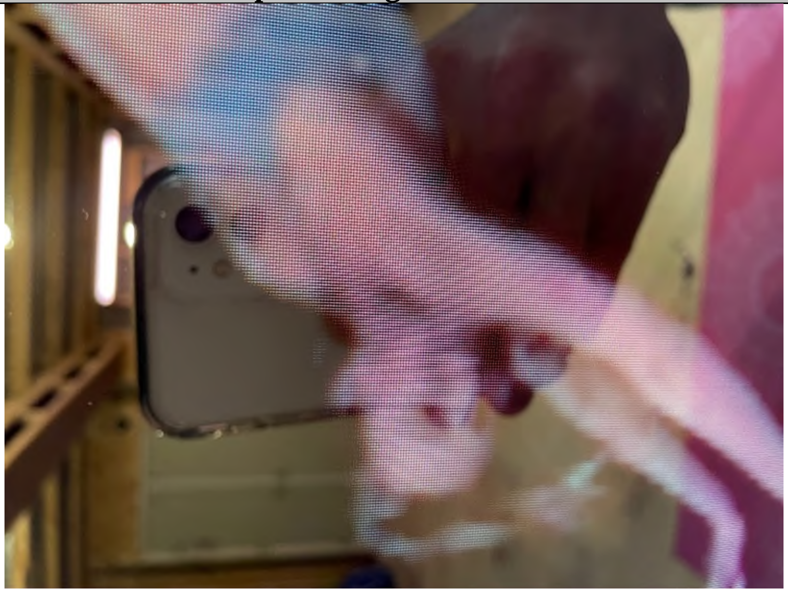
USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Other groups of streamlets are also available. For example, for the current test, bandwidth for the Mirror Device was constrained to 196Kbps, which caused the Mirror Device to display a "buffering" message while requesting and receiving a corresponding playlist and streamlets for a second bandwidth version of the live event video as shown below.

Claim	Claim Limitation	Example Infringement Evidence
		

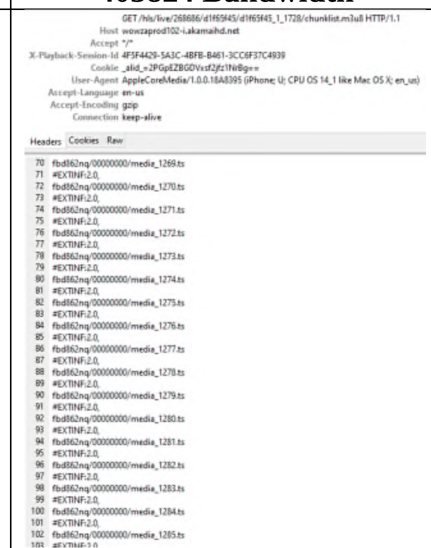
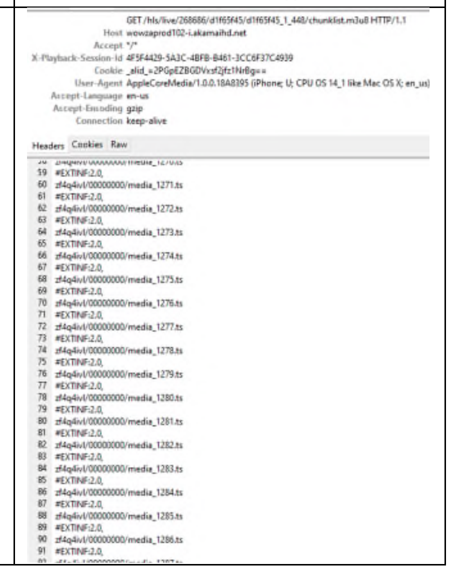
Claim	Claim Limitation	Example Infringement Evidence
		 The image resolution for the second bandwidth streamlet requested is noticeably lower quality as indicated by the pixelated edges of the instructor's body, as shown below.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		
	wherein the live event video is encoded at a plurality of different bitrates to create a plurality of streams including at least a low quality stream, a medium quality stream, and a high quality stream, each of the low quality stream, the medium quality stream, and the high quality stream	The "live event video is encoded at a plurality of different bitrates to create a plurality of streams." The plurality of different bitrates creates a plurality of streams "including at least a low quality stream, a medium quality stream, and a high quality stream." Further, "each of the low-quality stream, the medium-quality stream, and the high-quality stream compris[es] a group of streamlets encoded at the same respective one of the different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the live event video." As shown in the master playlist file, "**playlist.m3u8**," the video for the live event video is encoded at 6 different bitrates.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
	comprising a group of streamlets encoded at the same respective one of the different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the live event video;	#EXTM3U #EXT-X-VERSION:3 #EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080 ../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720 ../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480 ../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360 ../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288 ../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180 ../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 Filename: playlist.m3u8. The six different bitrates correspond to six different video streams of the same live event video but at varying quality: • 6434112 (referred to herein as "**6434112 Bandwidth**") • 864048 (referred to herein as "**864048 Bandwidth**") • 403824 (referred to herein as "**403824 Bandwidth**") • 367728 (referred to herein as "**367728 Bandwidth**") • 312832 (referred to herein as "**312832 Bandwidth**") • 249664 (referred to herein as "**249664 Bandwidth**") These different bitrate versions include at least a "low quality stream, a medium quality stream, and a high quality stream." For example, the **6434112 Bandwidth** version can be considered a high-quality stream, the **403824 Bandwidth** version can be considered a medium-quality stream, and the **249664 Bandwidth** version can be considered a low-quality stream. As shown herein, each of the high-quality stream (e.g., the **6434112 Bandwidth** stream), the medium-quality stream (e.g., the **403824 Bandwidth** stream), and the low-quality stream (e.g., the

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
249664 Bandwidth stream) comprise “a group of streamlets encoded at the same respective one of the different bitrates.” Each variant playlist includes at least two streamlets (e.g., “at least first and second streamlets”): a media_1274.ts” segment and a “media_1275.ts” segment. A comparison of the 6434112 Bandwidth, 403823 Bandwidth, and 249664 Bandwidth versions from above shows that each playlist includes segments with these file names. On information and belief, playlists for the other three variants also include these segments.
As discussed above, each streamlet corresponds to a portion of the live event video. Notably, for example, each bitrate version of the media_1275.ts segment has a duration of 2.0 seconds (as indicated in each line beginning with “#EXTINF”).
<table>
<tr><th>6434112 Bandwidth</th><th>403824 Bandwidth</th><th>249664 Bandwidth</th></tr>
<tr><td></td><td></td><td></td></tr>
</table>
Upon information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application. For example, during a test of the Mirror Devices, a first version, a second version, and a third version of the live video were captured. The first version corresponds to a high-
</td></tr>
</table>

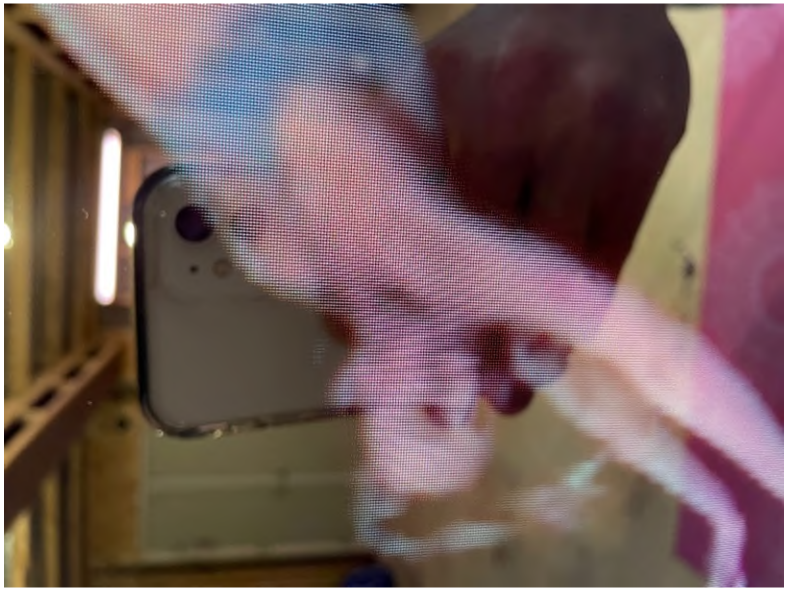
USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		quality stream, the second version corresponds to a medium-quality stream, and the third version corresponds to a low-quality stream. First version: 

USP 10,469,554 to Mirror

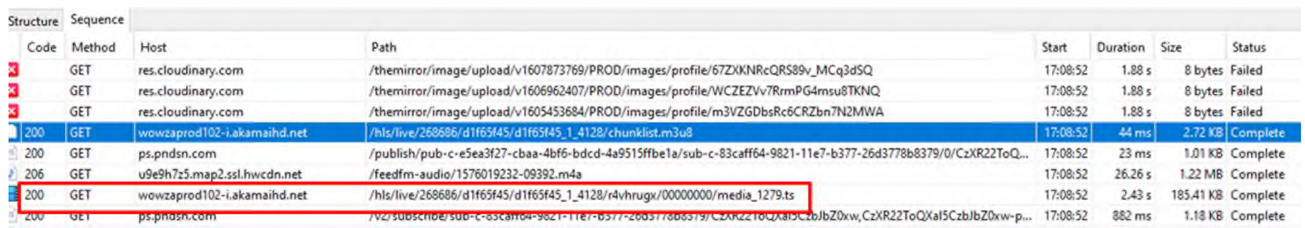
Claim	Claim Limitation	Example Infringement Evidence
		Second version: 

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Third version: 
	wherein at least one of the low-quality stream, the medium quality stream, and the high-quality stream is encoded at a bit rate of no less than 600 kbps; and	As shown above, "at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bit rate of no less than 600 kbps." At least the high-quality stream (**6326576 Bandwidth**) and one of the medium quality streams (**864048 Bandwidth**) is encoded at a bitrate of not less than 600 kbps as indicated by its "BANDWIDTH" attribute, which signals the upper bound of the overall bitrate for the streamlets in bits per second and is listed at over 6 megabits and 800 kilobits per second.
	wherein the first streamlets of each of the low quality stream, the medium	As shown above, the "first streamlets of each of the low quality stream, the medium quality stream, and the high quality stream each has an equal playback duration and each of the first streamlets encodes the same portion of the live event video at a different one of the different bitrates."

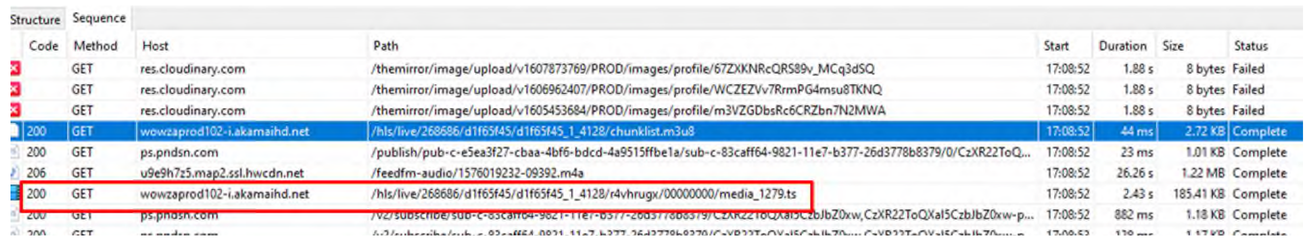
Claim	Claim Limitation	Example Infringement Evidence
	quality stream and the high quality stream each has an equal playback duration and each of the first streamlets encodes the same portion of the live event video at a different one of the different bitrates;	As discussed above, each of the **6434112 Bandwidth**, the **403824 Bandwidth**, and **249664 Bandwidth** variant playlists includes a "first streamlet" (e.g., **media_1275.ts** segment). Each of the variant "**media_1275.ts**" segments have an "equal playback duration" of 2.0 seconds (as indicated in each line beginning with "#EXTINF") and "encodes the same portion of the live event video" available in the Mirror Application in different bitrates. Upon information and belief, this is also true for the Mirror Devices as explained above.
	select a specific one of the low quality streams, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams;	The non-transitory machine readable instructions of the Mirror Application and the Mirror Devices cause the processor to "select a specific one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams." Based upon, at least in part, a determination of the available bandwidth, the Mirror Application and Mirror Devices may determine to "select a higher or lower bitrate version of the stream" and thereby "select a specific one of" the low-quality stream (e.g., the **249664 Bandwidth** stream), the medium-quality stream (e.g., the **403824 Bandwidth** stream), and the high-quality stream (e.g., the **6434112 Bandwidth** stream). As part of the testing, the Mirror Application was connected to the Internet through the Charles Proxy application. For the instant test, the Mirror Application selects the **403824 Bandwidth** stream as indicated by its request for a **403824 Bandwidth** playlist (*see* GET request for d1f65f45_1_1728/chunklist.m3u8) and subsequent request for the **403824 Bandwidth** version of the "media_1277.ts" file. When the available bandwidth was reduced during the test, the Mirror Application subsequently selected a different, lower bandwidth version of the stream. Below is an excerpt of the Charles Proxy application "Sequence" listing showing the same.

Method	Host	Path
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td colspan="3">..</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts</td></tr>
</table>
Upon information and belief, for at least the reasons stated above, the Mirror Devices and the Mirror Application operate in the same or substantially the same way.
</td></tr>
<tr><td></td><td>place a streamlet request to the server over the one or more network connections for the first streamlet of the selected stream;</td><td>
The non-transitory machine-readable instructions of the Mirror Application and Mirror Devices cause the processor to “place a streamlet request to the server over the one or more network connections for the first streamlet of the selected stream.”
For the instant test, the Mirror Application requests the 6434112 Bandwidth version of the “media_1279.ts” file. Below is an excerpt of the Charles “Sequence” listing showing the same.

Structure | Sequence
<table>
<tr><th>Code</th><th>Method</th><th>Host</th><th>Path</th><th>Start</th><th>Duration</th><th>Size</th><th>Status</th></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1607873769/PROD/images/profile/67ZXKNRcQRS89v_MCq3dSQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1606962407/PROD/images/profile/WCZEZVv7RrmPG4msu8TKNQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1605453684/PROD/images/profile/m3VZGDbsRc6CRZbn7N2MWA</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8</td><td>17:08:52</td><td>44 ms</td><td>2.72 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>/publish/pub-c-e5ea3f27-cbaa-4bf6-bdcd-4a9515ffbe1a/sub-c-83caff64-9821-11e7-b377-26d3778b8379/0/CzXR22ToQ...</td><td>17:08:52</td><td>23 ms</td><td>1.01 KB</td><td>Complete</td></tr>
<tr><td>206</td><td>GET</td><td>u9e9h7z5.map2.ssl.hwcdn.net</td><td>/feedfm-audio/1576019232-09392.m4a</td><td>17:08:52</td><td>26.26 s</td><td>1.22 MB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/r4vhrugx/00000000/media_1279.ts</td><td>17:08:52</td><td>2.43 s</td><td>185.41 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>/v2/subscribe/sub-c-83caff64-9821-11e7-b377-26d3778b8379/CzXR22ToQXal5CzbJbZ0xw,CzXR22ToQXal5CzbJbZ0xw-p...</td><td>17:08:52</td><td>882 ms</td><td>1.18 KB</td><td>Complete</td></tr>
</table>
Upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.
</td></tr>
<tr><td></td><td>receive the requested first streamlet from the server via the one</td><td>The non-transitory machine-readable instructions of the Mirror Application and Mirror Devices cause the processor to “receive the requested first streamlet from the server via the one or more network connections.”</td></tr>
</table>

USP 10,469,554 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td>or more network connections; and</td><td>For the instant test, the Mirror Application receives the 6434112 Bandwidth version of the “media_1279.ts” file. Below is an excerpt of the Charles “Sequence” listing showing the request is “complete.”.

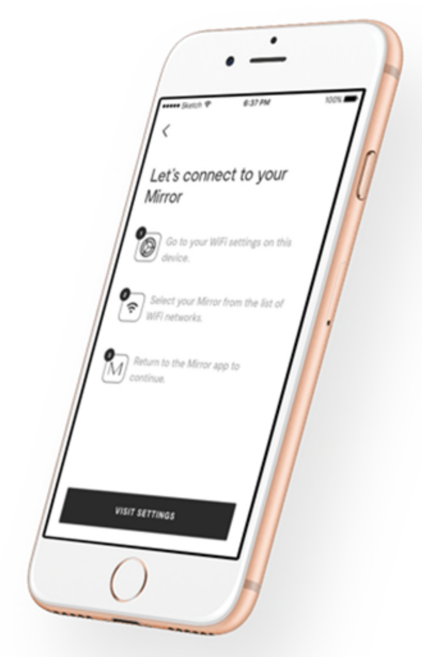

<table>
<tr><th>Code</th><th>Method</th><th>Host</th><th>Path</th><th>Start</th><th>Duration</th><th>Size</th><th>Status</th></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1607873769/PROD/images/profile/67ZXKNRcQRS89v_MCq3dSQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1606962407/PROD/images/profile/WCZEZVv7RrmPG4msu8TKNQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1605453684/PROD/images/profile/m3VZGDbsRc6CRZbn7N2MWA</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8</td><td>17:08:52</td><td>44 ms</td><td>2.72 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>/publish/pub-c-e5ea3f27-cbaa-4bf6-bdcd-4a9515ffbe1a/sub-c-83caff64-9821-11e7-b377-26d3778b8379/0/CzXR22ToQ...</td><td>17:08:52</td><td>23 ms</td><td>1.01 KB</td><td>Complete</td></tr>
<tr><td>206</td><td>GET</td><td>u9e9h7z5.map2.ssl.hwcdn.net</td><td>/feedfm-audio/1576019232-09392.m4a</td><td>17:08:52</td><td>26.26 s</td><td>1.22 MB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/r4vhrugx/00000000/media_1279.ts</td><td>17:08:52</td><td>2.43 s</td><td>185.41 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>...CzbJbZ0xw,CzXR22ToQXal5CzbJbZ0xw-p...</td><td>17:08:52</td><td>882 ms</td><td>1.18 KB</td><td>Complete</td></tr>
</table>

Upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.</td></tr>
<tr><td></td><td>provide the received first streamlet for playback of the live event video.</td><td>The non-transitory machine-readable instructions of the Mirror Application and Mirror Devices cause the processor to “provide the received first streamlet for playback of the live event video.”

As described above, the Mirror Application provides, or displays, the received 6434112 Bandwidth version of the “01279.ts” segment corresponding to the live event video on the screen of the end user station executing the Mirror Application or Mirror Device. In at least this way, upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.</td></tr>
<tr><td>30</td><td>A process executable by a content player device to stream a live event video over a network from a server for playback of the video by the content player device,</td><td>The Mirror Application is software that causes “a content player device to stream a live event video over a network from a server for playback of the video.” The Mirror Application is “executable by” end user stations that have a “content player device” and the Mirror Application streams of a selected live event video program for playback of the video. The streams are obtained over a network.

The images in this chart of the Mirror Application are from the Mirror Application running on an Apple iPhone XS (Mirror’s iOS Application). In addition, the Mirror Application is available to run on other devices. Unless otherwise noted, each of these devices is an end user station having a “content player device.”</td></tr>
</table>

USP 10,469,554 to Mirror

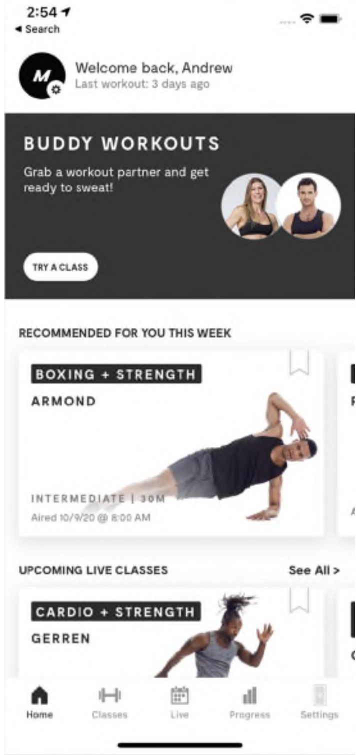
Claim	Claim Limitation	Example Infringement Evidence
	the process comprising:	**MIRROR DIGITAL OVERVIEW** Access all of MIRROR's Live and On Demand Classes 24/7 from your phone, tablet, and smart TV devices. This feature is included for free with the MIRROR Membership and access is available immediately (whether you already have your Mirror or are awaiting delivery). Please note, it is not currently possible to purchase MIRROR Digital if you do not own a Mirror. https://mirror.kustomer.help/en_us/mirror-digital-overview-B1pnVd8UU **GET THE MIRROR APP** To get started setting up your Mirror, you need to download the MIRROR app from the Apple App Store or Google Play Store. The app will take you through everything you need to know. Available on the App Store Get it on Google play Need help? Email us at hello@mirror.co  https://www.mirror.co/app.

USP 10,469,554 to Mirror

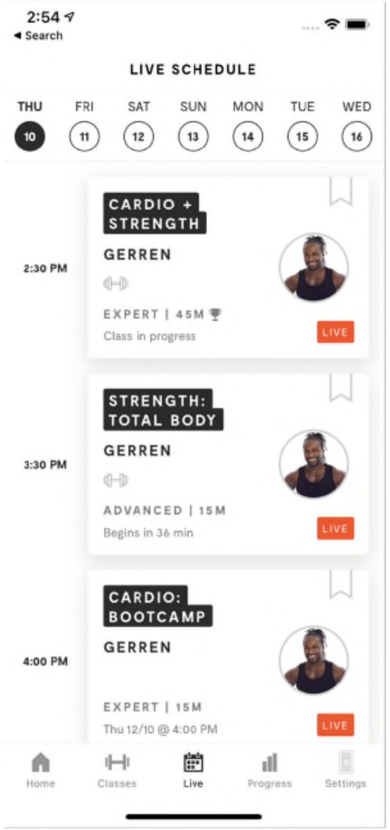
Claim	Claim Limitation	Example Infringement Evidence
		MIRROR APP The MIRROR App allows you to access and customize the Mirror experience. The MIRROR App is available for both iOS and Android! • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-app--S1dDC_tYm.

USP 10,469,554 to Mirror

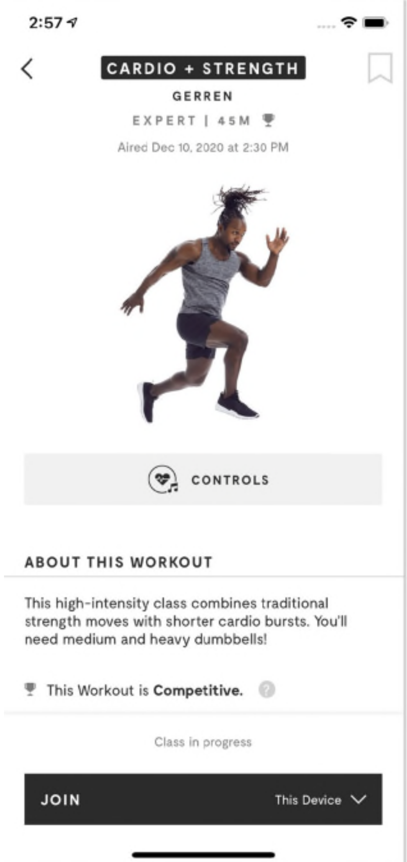
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Apple App Store When launched, the Mirror Application displays a main menu:

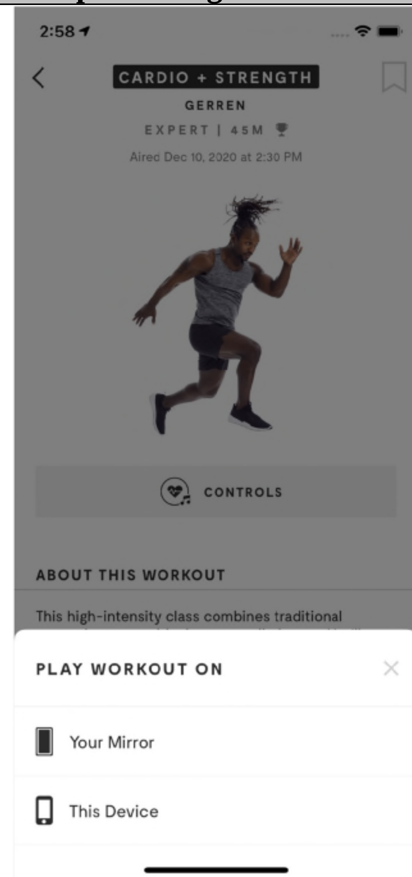
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application The main menu of the Mirror Application displays classes that are each an "event video." The "Upcoming Live Classes" section of the Mirror Application main menu displays a preview of ongoing and upcoming live event video programming.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting an ongoing live class from the list causes the Mirror Application to display more details regarding the class and provides the user with the option to join the class.

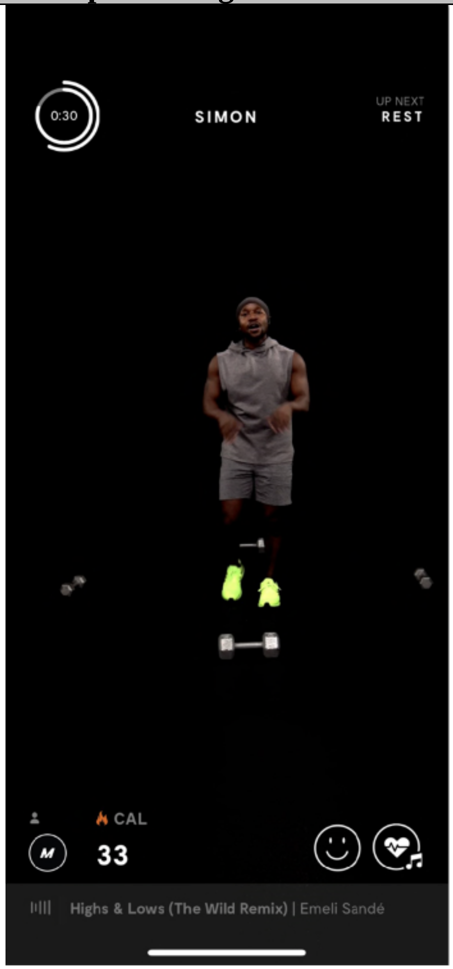
USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting the "Join" button for an ongoing live event causes the Mirror Application to provide options to stream the class to a variety of end user stations over the Internet, including the iOS device that the user is using or the separate Mirror device.

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting “Your Mirror” causes the selected live event video and other materials to be streamed on the user’s Mirror device, which is connected to the Internet.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Alternatively, selecting "This Device" causes the selected workout video and other materials to be streamed on the user's iOS device:

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application As shown above, Mirror Devices include content player devices to stream a video over a network from a server for playback of the video. The Mirror Devices obtain streams of a selected video program for playback. The streams are obtained from the Mirror Server(s) over a network.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
	establishing one or more network connections between the content player device and the server, wherein the server accesses a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the live event video;	The Mirror Application and Mirror Devices perform the step of "establishing one or more network connections between the content player device and the server" that "accesses a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates," where each group comprises "at least first and second streamlets" and "each of the streamlets corresponding to a portion of the live event video." The Mirror Application requires an internet connection. **PRELOAD CLASSES ON MIRROR DIGITAL** You currently cannot preload classes on the MIRROR Digital, however this feature is coming soon! If you are not able to connect WiFi or are in a tough WiFi environment, you can always use cellular data to stream classes on the MIRROR App. Please consult your cell phone provider for questions about data usage and your plan. https://mirror.kustomer.help/en_us/preload-classes-on-mirror-app-H12XPdUUL. To stream a live event video, such as that shown above, the Mirror Application requests a stream of a selected live event video program via a network connection . When the Mirror Application accesses a selected live event video program, it requests and receives a playlist file that shows the available versions of the live event program at different bandwidths and resolutions. For the following test, a live event video was selected. In the test, the Mirror Device makes an HTTPS GET request for a master playlist named "**playlist.m3u8**" that specifies the available streams and provides links to the playlists for those streams. The following master playlist named "**playlist.m3u8**" is returned.

Claim	Claim Limitation	Example Infringement Evidence

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080
../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720
../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480
../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360
../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288
../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180
../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
```

Filename: playlist.m3u8.

This is a master playlist file according to the HLS specification.[2] The playlist shows six versions of the stream, denoted by each #EXT-X-STREAM-INF tag at the following bandwidth:

- 6434112 (referred to herein as "**6434112 Bandwidth**")
- 864048 (referred to herein as "**864048 Bandwidth**")
- 403824 (referred to herein as "**403824 Bandwidth**")
- 367728 (referred to herein as "**367728 Bandwidth**")
- 312832 (referred to herein as "**312832 Bandwidth**")
- 249664 (referred to herein as "**249664 Bandwidth**")

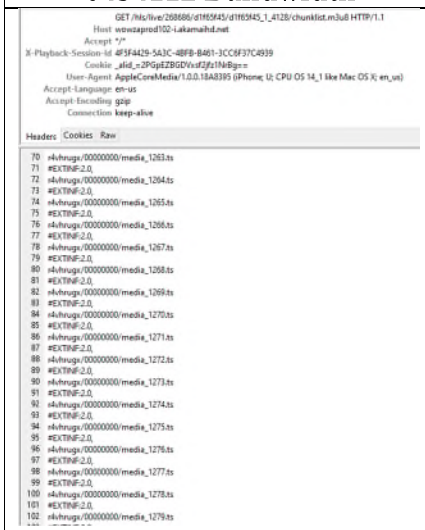
For each of these versions, the master playlist provides a link to a playlist file for the specified version of the selected live event video at a particular bandwidth and resolution, which is called a "variant" in HLS.

[2] RFC 8216 (HLS Live Streaming), Section 4.3.4 (Master Playlist Tags)

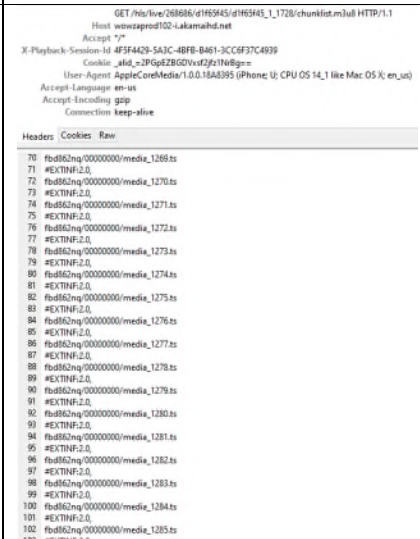
USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		These different bitrate versions include at least a "low quality stream, a medium quality stream, and a high quality stream." For example, the **6434112 Bandwidth** version can be considered a high-quality stream, the **403824 Bandwidth** version can be considered a medium-quality stream, and the **249664 Bandwidth** version can be considered a low-quality stream. As shown herein, each of the high-quality stream (e.g., the **6434112 Bandwidth** stream), the medium-quality stream (e.g., the **403824 Bandwidth** stream), and the low-quality stream (e.g., the **249664 Bandwidth** stream) comprise "a group of streamlets encoded at the same respective one of the different bitrates." Each variant playlist includes at least two streamlets (e.g., "at least first and second streamlets"): a **media_1238.ts**" segment and a "**media_1239.ts**" segment. A comparison of the **6434112 Bandwidth**, **403823 Bandwidth**, and **249664 Bandwidth** versions from above shows that each playlist includes segments with these file names. On information and belief, playlists for the other three variants also include these segments.

6434112 Bandwidth	403824 Bandwidth	249664 Bandwidth



```
GET /hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 HTTP/1.1
Host wowzaprod102-i.akamaihd.net
Accept */*
X-Playback-Session-Id 4F5F4429-5A3C-4BFB-B461-3CC6F37C4939
Cookie _alid_=2PGpEZBGDVxsf2jfz1NrBg==
User-Agent AppleCoreMedia/1.0.0.18A8395 (iPhone; U; CPU OS 14_1 like Mac OS X; en_us)
Accept-Language en-us
Accept-Encoding gzip
Connection keep-alive
Headers Cookies Raw
r4vhrugx/00000000/media_1263.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1264.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1265.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1266.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1267.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1268.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1269.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1270.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1271.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1272.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1273.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1274.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1275.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1276.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1277.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1278.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1279.ts
```



```
GET /hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 HTTP/1.1
Host wowzaprod102-i.akamaihd.net
Accept */*
X-Playback-Session-Id 4F5F4429-5A3C-4BFB-B461-3CC6F37C4939
Cookie _alid_=2PGpEZBGDVxsf2jfz1NrBg==
User-Agent AppleCoreMedia/1.0.0.18A8395 (iPhone; U; CPU OS 14_1 like Mac OS X; en_us)
Accept-Language en-us
Accept-Encoding gzip
Connection keep-alive
Headers Cookies Raw
fbd862nq/00000000/media_1269.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1270.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1271.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1272.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1273.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1274.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1275.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1276.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1277.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1278.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1279.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1280.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1281.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1282.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1283.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1284.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1285.ts
#EXTINF:2.0
```

```
GET /hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 HTTP/1.1
Host wowzaprod102-i.akamaihd.net
Accept */*
X-Playback-Session-Id 4F5F4429-5A3C-4BFB-B461-3CC6F37C4939
Cookie _alid_=2PGpEZBGDVxsf2jfz1NrBg==
User-Agent AppleCoreMedia/1.0.0.18A8395 (iPhone; U; CPU OS 14_1 like Mac OS X; en_us)
Accept-Language en-us
Accept-Encoding gzip
Connection keep-alive
Headers Cookies Raw
zf4q4ivl/00000000/media_1270.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1271.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1272.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1273.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1274.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1275.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1276.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1277.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1278.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1279.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1280.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1281.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1282.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1283.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1284.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1285.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1286.ts
#EXTINF:2.0,
```

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
As discussed above, each streamlet corresponds to a portion of the live event video. Notably, for example, each bitrate version of the media_1275.ts segment has a duration of 2.0 seconds (as indicated in each line beginning with “#EXTINF”).
The Mirror Application initially selects the 6434112 Bandwidth (1080p – high bandwidth) version of the stream and makes a request for the corresponding variant playlist file named “chunklist.m3u8.” That file is returned with the following contents (a portion of which is shown below).
<pre>
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1232
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:08.356Z
#EXTINF:2.0,
r4vhrugx/00000000/media_1232.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1233.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1234.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1235.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1236.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1237.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1238.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1239.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1240.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1241.ts
#EXTINF:2.0,
</pre>
</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		Path: https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 As noted above, the variant playlist file is an HLS playlist. The variant playlist file identifies a plurality of segments or "streamlets" that are part of the **6434112 Bandwidth** group of streamlets. Each line in the file "**chunklist.m3u8**" that begins with "#EXTINF" specifies the length of the segments in seconds (2.0). The line below the #EXTINF entry is the relative location of the video file for the segment (e.g., r4vhrugx/0000000/media_1232.ts). The Mirror Application uses HTTPS GET requests to retrieve the segments of the encoded live event video specified in the file above The Mirror Application makes a request for a segment media_1232.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected live program. As long as the viewer stays on the selected live event video and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **6434112 Bandwidth** version) . While the example above relates to the **6434112 Bandwidth** group of streamlets, other groups of streamlets are also available. If the available bandwidth for the network connection decreases, for example, the Mirror Application will continue to request segments to continue streaming the live event program, but at one of the lower bandwidths. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variable playlist file for the **403824 Bandwidth** named "chunklist.m3u8." That file is returned with the following contents showing a portion of the **403824 Bandwidth** group of segments for the live event video being streamed.

USP 10,469,554 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td><pre>
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1238
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z
#EXTINF:2.0,
fbd862nq/00000000/media_1238.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1239.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1240.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1241.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1242.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1243.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1244.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1245.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1246.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1247.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1248.ts
</pre>
Filename: chunklist.m3u8

The Mirror Application then makes the request for media_1238.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected live event video program at the 403824 Bandwidth. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror Application will continue to</td></tr>
</table>

USP 10,469,554 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>

request and receive playlists corresponding to the current, chosen resolution (in this case, the **403824 Bandwidth** version).

As the bandwidth is further constrained, the Mirror Application makes another request for a lower quality stream. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variable playlist file for the **249664 Bandwidth** named “chunklist.m3u8.” That file is returned with the following contents showing a portion of the **249664 Bandwidth** group of segments for the live event video being streamed.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1241
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:26.354Z
#EXTINF:2.0,
zf4q4ivl/00000000/media_1241.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1242.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1243.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1244.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1245.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1246.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1247.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1248.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1249.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1250.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1251.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1252.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1253.ts
```

</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		File: **chunklist.m3u8** The Mirror Application then makes the request for media_1281.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected live event video program at the **249664 Bandwidth**. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **249664 Bandwidth** version). A portion of a subsequently retrieved playlist's contents are shown below.

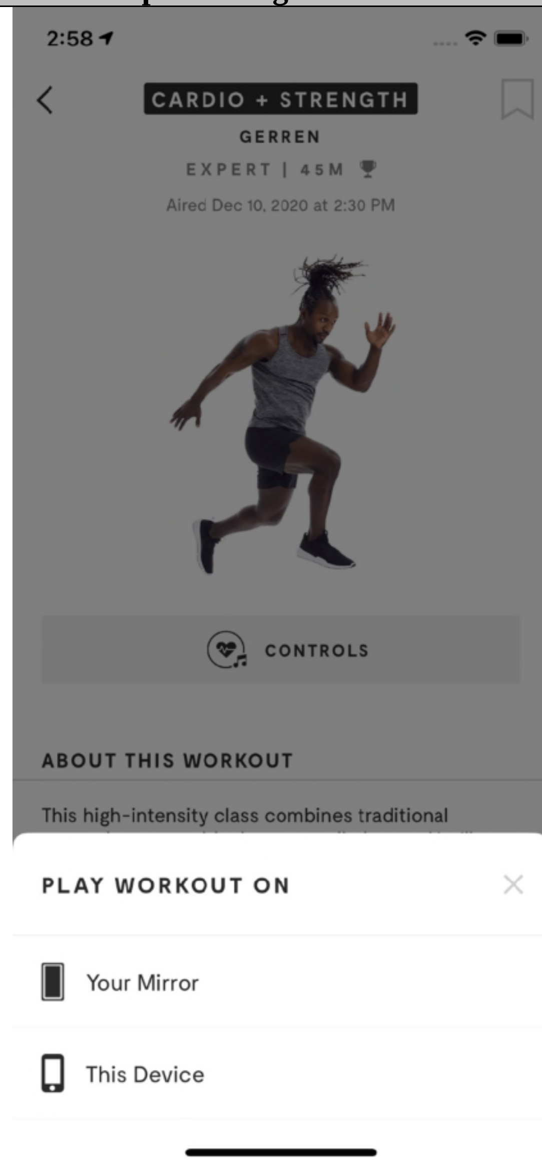
Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1245 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:34.354Z #EXTINF:2.0, zf4q4ivl/00000000/media_1245.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1246.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1247.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1248.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1249.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1250.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1251.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1252.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1253.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1254.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1255.ts #EXTINF:2.0, Filename: **chunklist.m3u8** The subsequently retrieved playlist includes additional video segments that were not included in the previous playlist file, for example: "**media_1254.ts**" and "**media_1255.ts.**" The Mirror Application continues to request, receive, and playback successive segments of the live event video to stream the live event video.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
The Mirror Devices also require that users provide an internet connection.

CONNECTION

<table>
<tr><td>INTERNET</td><td>Dual-band 802.11 A/B/G/N Wi-Fi</td></tr>
<tr><td>APP</td><td>Controlled by iOS or Android companion app</td></tr>
<tr><td>HEART RATE</td><td>Syncs with Bluetooth™ heart rate monitors, Apple Watches, and Android Wear OS Watches</td></tr>
<tr><td>AUDIO</td><td>Pairs with Bluetooth™ speakers and headphones</td></tr>
</table>
https://www.mirror.co/shop/mirror

To stream a live event video, such as that shown above, the Mirror Device requests a stream of a selected live event video via a network connection . The iOS application provides the interface for interacting with a Mirror Device (i.e., selecting a live or on demand class to stream). After selecting a class, the user selects whether to use the iOS device or Mirror Device to view the content (i.e., stream and participate in the workout).
</td></tr>
</table>

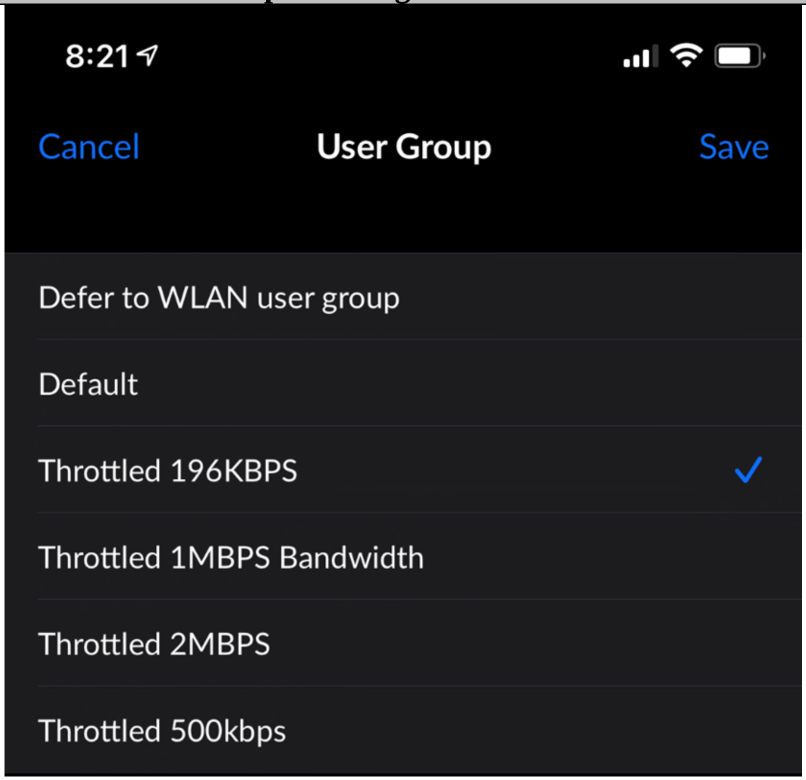
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Selecting "Your Mirror" causes the Mirror Device to initiate streaming requests:  For the following test, a live programming event was selected. Based on the test, and upon information and belief, the Mirror Devices operates in the same or substantially the same way as the Mirror Application. For example, when the Mirror Device(s) accesses a selected live event video, the Mirror Device(s) initially selects a first bandwidth version of the stream, makes a request for the segments of the group corresponding to the selected bandwidth version of the live event program, receives segments from the group corresponding to the selected bandwidth version, and then plays the requested and received segments on the Mirror Device content player as shown below.

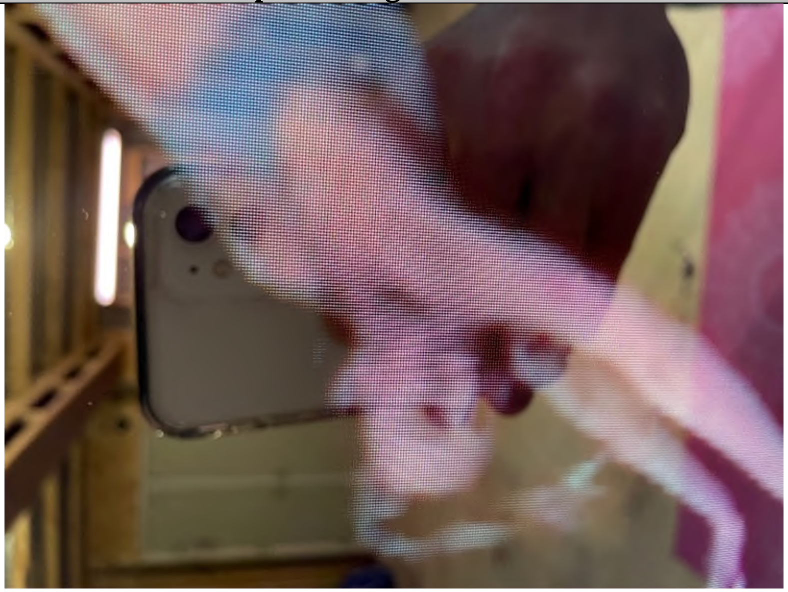
USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Other groups of streamlets are also available. For example, for the current test, bandwidth for the Mirror Device was constrained to 196Kbps, which caused the Mirror Device to display a "buffering" message while requesting and receiving a corresponding playlist and streamlets for a second bandwidth version of the live event video as shown below.

Claim	Claim Limitation	Example Infringement Evidence
		

Claim	Claim Limitation	Example Infringement Evidence
		 The image resolution for the second bandwidth streamlet requested is noticeably lower quality as indicated by the pixelated edges of the instructor's body, as shown below.

USP 10,469,554 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		
	wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and	As shown above, "at least one of the low-quality streams, the medium-quality streams, and the high-quality streams is encoded at a bit rate of no less than 600 kbps." At least the high-quality stream (**6326576 Bandwidth**) and one of the medium quality streams (**864048 Bandwidth**) is encoded at a bitrate of not less than 600 kbps as indicated by its "BANDWIDTH" attribute, which signals the upper bound of the overall bitrate for the streamlets in bits per second and is listed at over 6 megabits and 800 kilobits per second.
	wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the	As shown above, the "first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the live event video in the low quality stream, the medium quality stream, and the high quality stream," and "the first streamlet of the low quality stream ha[s] a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream."

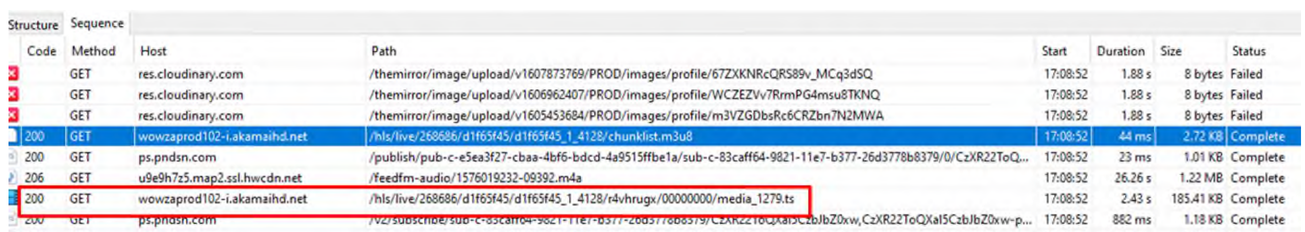
Claim	Claim Limitation	Example Infringement Evidence
	live event video in the low quality stream, the medium quality stream, and the high quality stream, the first streamlet of the low quality stream having a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream;	As discussed above, each of the **6434112 Bandwidth**, the **403824 Bandwidth**, and **249664 Bandwidth** variant playlists includes a "first streamlet" (e.g., **media_1275.ts** segment). Each of the variant "**media_1275.ts**" segments has "the same first duration" of 2.0 seconds (as indicated in each line beginning with "#EXTINF") and "encodes the same first portion of the live event video" identified by the "media_1275.ts" segment in different bitrates. Upon information and belief, this is also true for the Mirror Devices as explained above.
	selecting, by the content player device, a currently selected one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the live event video;	The Mirror Application and the Mirror Devices perform the step of "selecting, by the content player device, a currently selected one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the live event video." Based upon, at least in part, the bandwidth available, the Mirror Application and Mirror Devices may determine to "select a higher or lower bitrate version of the stream" and thereby "select a specific one of" the low-quality stream (e.g., the **249664 Bandwidth** stream), the medium-quality stream (e.g., the **403824 Bandwidth** stream), and the high-quality stream (e.g., the **6434112 Bandwidth** stream). As part of the testing, the Mirror Application was connected to the Internet through the Charles Proxy application. For the instant test, the Mirror Application selects the **403824 Bandwidth** stream as indicated by its request for a **403824 Bandwidth** playlist (*see* GET request for d1f65f45_1_1728/chunklist.m3u8) and subsequent request for the **403824 Bandwidth** version of the "media_1277.ts" file. When the available bandwidth was reduced during the test, the Mirror Application subsequently selected a different, lower bandwidth version of the stream. Below is an excerpt of the Charles Proxy application "Sequence" listing showing the same.

USP 10,469,554 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8</td></tr>
<tr><td colspan="3">..</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts</td></tr>
</table>

Upon information and belief, for at least the reasons stated above, the Mirror Devices and the Mirror Application operate in the same or substantially the same way.
</td></tr>
<tr><td></td><td>placing a streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the live event video;</td><td>
The Mirror Application and Mirror Devices perform the step of “placing a streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the live event video.”

For the instant test, the Mirror Application requests the 6434112 Bandwidth version of the “media_1279.ts” file. Below is an excerpt of the Charles “Sequence” listing showing the same.



Structure | Sequence
<table>
<tr><th>Code</th><th>Method</th><th>Host</th><th>Path</th><th>Start</th><th>Duration</th><th>Size</th><th>Status</th></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1607873769/PROD/images/profile/67ZXKNRcQRS89v_MCq3dSQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1606962407/PROD/images/profile/WCZEZVv7RrmPG4msu8TKNQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1605453684/PROD/images/profile/m3VZGDbsRc6CRZbn7N2MWA</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8</td><td>17:08:52</td><td>44 ms</td><td>2.72 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>/publish/pub-c-e5ea3f27-cbaa-4bf6-bdcd-4a9515ffbe1a/sub-c-83caff64-9821-11e7-b377-26d3778b8379/0/CzXR22ToQ...</td><td>17:08:52</td><td>23 ms</td><td>1.01 KB</td><td>Complete</td></tr>
<tr><td>206</td><td>GET</td><td>u9e9h7z5.map2.ssl.hwcdn.net</td><td>/feedfm-audio/1576019232-09392.m4a</td><td>17:08:52</td><td>26.26 s</td><td>1.22 MB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/r4vhrugx/00000000/media_1279.ts</td><td>17:08:52</td><td>2.43 s</td><td>185.41 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>/v2/subscribe/sub-c-83caff64-9821-11e7-b377-26d3778b8379/CzXR22ToQXal5CzbJbZ0xw,CzXR22ToQXal5CzbJbZ0xw-p...</td><td>17:08:52</td><td>882 ms</td><td>1.18 KB</td><td>Complete</td></tr>
</table>

Upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.
</td></tr>
</table>

USP 10,469,554 to Mirror

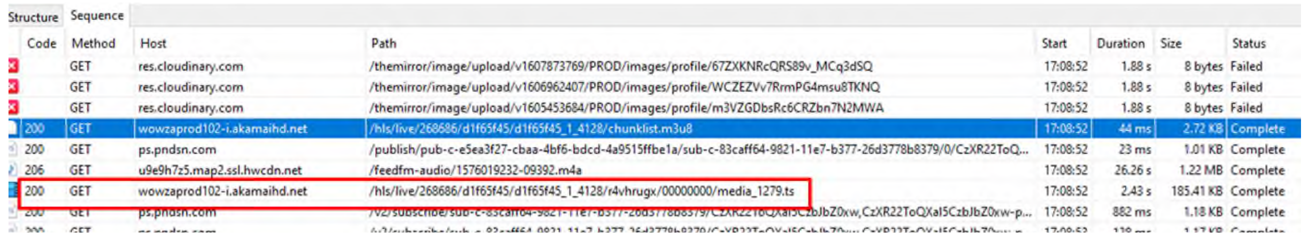
Claim	Claim Limitation	Example Infringement Evidence
	receiving the requested streamlet from the server via the one or more network connections; and	The Mirror Application and Mirror Devices perform the step of “receiving the requested streamlet from the server via the one or more network connections.” For the instant test, the Mirror Application receives the **6434112 Bandwidth** version of the “media_1279.ts” file. Below is an excerpt of the Charles “Sequence” listing showing the request is “complete.”  Upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.
	rendering, by the content player device, the received streamlet for playback of the live event video.	The device running the Mirror Application, and likewise the Mirror Devices, perform the step of “rendering, by the content player device, the received streamlet for playback of the live event video.” As shown above, the Mirror Application provides, or displays, the received **6434112 Bandwidth** version of the “01286.ts” file corresponding to the live event video. Upon information and belief, the Mirror Devices operate in the same or substantially the same way, for at least the reasons shown above.

EXHIBIT C

US010469555B2

(12) **United States Patent**
Brueck et al.

(10) **Patent No.: US 10,469,555 B2**
(45) **Date of Patent: *Nov. 5, 2019**

(54) **APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING**

(71) Applicant: **DISH Technologies L.L.C.**, Englewood, CO (US)

(72) Inventors: **David F. Brueck**, Saratoga Springs, UT (US); **Mark B. Hurst**, Cedar Hills, UT (US); **R. Drew Major**, Orem, UT (US)

(73) Assignee: **DISH Technologies L.L.C.**, Englewood, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **16/252,356**

(22) Filed: **Jan. 18, 2019**

(65) **Prior Publication Data**

US 2019/0158561 A1 May 23, 2019

Related U.S. Application Data

(63) Continuation of application No. 16/252,188, filed on Jan. 18, 2019, which is a continuation of application

(Continued)

(51) **Int. Cl.**
H04L 29/06 (2006.01)
H04L 12/927 (2013.01)

(Continued)

(52) **U.S. Cl.**
CPC ***H04L 65/607*** (2013.01); ***G06F 16/183*** (2019.01); ***G06F 16/71*** (2019.01);

(Continued)

(58) **Field of Classification Search**
CPC H04N 19/34; H04N 19/40; H04N 21/2662; H04N 21/234327; H04N 21/2393;

(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,535,355	A	8/1985	Arn et al.
5,168,356	A	12/1992	Acampora et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CA	2466482	A1	5/2003
EP	0919952	A1	6/1999

(Continued)

OTHER PUBLICATIONS

Roy, S., et al., "Architecture of a Modular Streaming Media Server for Content Delivery Networks," 2002 IEEE. Published in the 2003 International Conference on Multimedia and Expo ICME 2003.

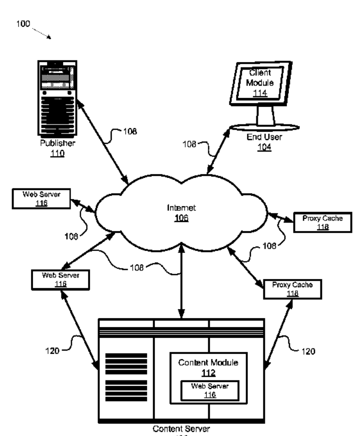
(Continued)

Primary Examiner — Chirag R Patel
(74) *Attorney, Agent, or Firm* — Lorenz & Kopf LLP

(57) **ABSTRACT**

An apparatus for multi-bitrate content streaming includes a receiving module configured to capture media content, a streamlet module configured to segment the media content and generate a plurality of streamlets, and an encoding module configured to generate a set of streamlets. The system includes the apparatus, wherein the set of streamlets comprises a plurality of streamlets having identical time indices and durations, and each streamlet of the set of streamlets having a unique bitrate, and wherein the encoding module comprises a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid. A method includes receiving media content, segmenting the media content and generating a plurality of streamlets, and generating a set of streamlets.

27 Claims, 11 Drawing Sheets



Related U.S. Application Data

No. 16/004,056, filed on Jun. 8, 2018, which is a continuation of application No. 15/414,025, filed on Jan. 24, 2017, now Pat. No. 9,998,516, which is a continuation of application No. 14/719,122, filed on May 21, 2015, now Pat. No. 9,571,551, which is a continuation of application No. 14/106,051, filed on Dec. 13, 2013, now Pat. No. 9,071,668, which is a continuation of application No. 13/617,114, filed on Sep. 14, 2012, now Pat. No. 8,612,624, which is a continuation of application No. 12/906,940, filed on Oct. 18, 2010, now Pat. No. 8,402,156, which is a continuation-in-part of application No. 11/673,483, filed on Feb. 9, 2007, now Pat. No. 7,818,444, which is a continuation-in-part of application No. 11/116, 783, filed on Apr. 28, 2005, now Pat. No. 8,868,772.

(60) Provisional application No. 60/566,831, filed on Apr. 30, 2004.

(51) **Int. Cl.**

H04L 12/801	(2013.01)
G06F 16/71	(2019.01)
G06F 16/182	(2019.01)
H04N 7/24	(2011.01)
H04N 21/2343	(2011.01)
H04N 21/433	(2011.01)
H04N 21/84	(2011.01)
H04N 21/845	(2011.01)
H04L 29/08	(2006.01)
H04N 21/2662	(2011.01)

(52) **U.S. Cl.**
CPC ***H04L 29/06027*** (2013.01); ***H04L 47/12*** (2013.01); ***H04L 47/801*** (2013.01); ***H04L 65/1069*** (2013.01); ***H04L 65/4069*** (2013.01); ***H04L 65/608*** (2013.01); ***H04L 65/80*** (2013.01); ***H04L 67/02*** (2013.01); ***H04L 67/2842*** (2013.01); ***H04L 67/32*** (2013.01); ***H04N 7/24*** (2013.01); ***H04N 21/23439*** (2013.01); ***H04N 21/2662*** (2013.01); ***H04N 21/4331*** (2013.01); ***H04N 21/84*** (2013.01); ***H04N 21/8456*** (2013.01)

(58) **Field of Classification Search**
CPC . H04L 65/80; H04L 67/2842; H04L 65/4069; H04L 65/607; H04L 65/608
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,267,334	A	11/1993	Normille et al.	
5,404,446	A	4/1995	Bowater et al.	
5,687,095	A	11/1997	Haskell et al.	
5,732,183	A	3/1998	Sugiyama	
5,768,527	A	6/1998	Zhu et al.	
5,812,786	A *	9/1998	Seazholtz	H04M 11/062 370/465
5,841,432	A	11/1998	Carmel et al.	
5,953,506	A	9/1999	Kalra et al.	
6,091,775	A	7/2000	Hibi et al.	
6,091,777	A	7/2000	Guetz et al.	
6,122,660	A	9/2000	Baransky et al.	
6,185,736	B1	2/2001	Ueno	
6,195,680	B1	2/2001	Goldszmidt et al.	
6,366,614	B1	4/2002	Pian et al.	
6,374,289	B2	4/2002	Delaney et al.	
6,389,473	B1	5/2002	Carmel et al.	
6,449,719	B1	9/2002	Baker	
6,486,803	B1	11/2002	Luby et al.	
6,490,627	B1	12/2002	Kalra et al.	
6,496,980	B1 *	12/2002	Tillman	H04N 7/17318 348/E7.071
6,510,553	B1	1/2003	Hazra	
6,574,591	B1	6/2003	Kleiman et al.	
6,604,118	B2	8/2003	Klleiman et al.	
6,618,752	B1	9/2003	Moore et al.	
6,708,213	B1	3/2004	Bommaiah et al.	
6,721,723	B1	4/2004	Gibson et al.	
6,731,600	B1	5/2004	Patel et al.	
6,757,796	B1	6/2004	Hofmann	
6,760,772	B2	7/2004	Zou et al.	
6,795,863	B1	9/2004	Doty, Jr.	
6,845,107	B1	1/2005	Kitazawa et al.	
6,850,965	B2	2/2005	Allen	
6,859,839	B1	2/2005	Zahorjan et al.	
6,874,015	B2	3/2005	Kaminsky et al.	
6,968,387	B2	11/2005	Lanphear	
6,976,090	B2	12/2005	Ben-Shaul et al.	
7,054,365	B2	5/2006	Kim et al.	
7,054,774	B2	5/2006	Batterberry et al.	
7,054,911	B1	5/2006	Lango et al.	
7,075,986	B2	7/2006	Girod et al.	
7,093,001	B2	8/2006	Yang et al.	
7,096,271	B1	8/2006	Omoigui et al.	
7,099,954	B2	8/2006	Li et al.	
7,116,894	B1	10/2006	Chatterton	
7,174,385	B2	2/2007	Li	
7,194,549	B1	3/2007	Lee et al.	
7,240,100	B1	7/2007	Wein et al.	
7,260,640	B1	8/2007	Kramer et al.	
7,274,740	B2	9/2007	van Beek et al.	
7,295,520	B2	11/2007	Lee et al.	
7,310,678	B2	12/2007	Gunaseelan et al.	
7,325,073	B2	1/2008	Shao et al.	
7,328,243	B2	2/2008	Yaeger et al.	
7,330,908	B2	2/2008	Jungek	
7,334,044	B1	2/2008	Allen	
7,349,358	B2	3/2008	Hennessey et al.	
7,349,976	B1	3/2008	Glaser et al.	
7,369,610	B2 *	5/2008	Xu	H04N 21/2662 375/240.08
7,376,747	B2	5/2008	Hartop	
7,391,717	B2	6/2008	Kiemets et al.	
7,408,984	B2	8/2008	Lu et al.	
7,412,531	B1	8/2008	Lango et al.	
7,477,688	B1	1/2009	Zhang et al.	
7,523,181	B2	4/2009	Swildens et al.	
7,536,469	B2	5/2009	Chou et al.	
7,546,355	B2	6/2009	Kalnitsky	
7,558,869	B2	7/2009	Leon et al.	
7,577,750	B2	8/2009	Shen et al.	
7,593,333	B2	9/2009	Li et al.	
7,599,307	B2	10/2009	Seckin et al.	
7,609,652	B2	10/2009	Kellerer et al.	
7,653,735	B2	1/2010	Mandato et al.	
7,707,303	B2	4/2010	Albers et al.	
7,719,985	B2	5/2010	Lee et al.	
7,760,801	B2	7/2010	Ghanbari et al.	
7,779,135	B2	8/2010	Hudson et al.	
7,788,395	B2	8/2010	Bowra et al.	
7,797,439	B2	9/2010	Cherkasova et al.	
7,817,985	B2	10/2010	Moon	
7,818,444	B2	10/2010	Brueck et al.	
7,925,781	B1	4/2011	Chan et al.	
7,934,159	B1 *	4/2011	Rahman	H04N 21/4825 715/716
8,036,265	B1	10/2011	Reynolds et al.	
8,370,514	B2	2/2013	Hurst et al.	
8,402,156	B2	3/2013	Brueck et al.	
8,521,836	B2	8/2013	Kewalramani et al.	
8,612,624	B2	12/2013	Brueck et al.	
8,683,066	B2	3/2014	Hurst et al.	
8,686,066	B2	4/2014	Kwampian et al.	
8,868,772	B2	10/2014	Major et al.	
8,880,721	B2	11/2014	Hurst et al.	
9,344,496	B2	5/2016	Hurst et al.	
9,462,074	B2	10/2016	Guo et al.	

(56) **References Cited**

U.S. PATENT DOCUMENTS

2001/0013128 A1	8/2001	Hagai et al.	
2001/0047423 A1	11/2001	Shao et al.	
2002/0029274 A1	3/2002	Allen	
2002/0073167 A1	6/2002	Powell et al.	
2002/0091840 A1	7/2002	Pulier et al.	
2002/0097750 A1	7/2002	Gunaseelan et al.	
2002/0131496 A1	9/2002	Vasudevan et al.	
2002/0144276 A1	10/2002	Radford et al.	
2002/0152317 A1	10/2002	Wang et al.	
2002/0152318 A1	10/2002	Menon et al.	
2002/0156912 A1	10/2002	Hurst et al.	
2002/0161898 A1	10/2002	Hartop et al.	
2002/0161908 A1	10/2002	Benitez et al.	
2002/0161911 A1	10/2002	Pinckney, III et al.	
2002/0169926 A1	11/2002	Pinckney, III et al.	
2002/0174434 A1	11/2002	Lee et al.	
2002/0176418 A1	11/2002	Hunt et al.	
2002/0178330 A1	11/2002	Schlowsky-Fischer et al.	
2002/0188745 A1	12/2002	Hughes et al.	
2003/0005455 A1	1/2003	Bowers	
2003/0014684 A1	1/2003	Kashyap	
2003/0018966 A1	1/2003	Cook et al.	
2003/0021166 A1	1/2003	Soloff	
2003/0021282 A1	1/2003	Hospodor	
2003/0023982 A1*	1/2003	Lee	H04N 21/234327 725/116
2003/0055995 A1	3/2003	Ala Honkola	
2003/0065803 A1	4/2003	Heuvelman	
2003/0067872 A1	4/2003	Harrell et al.	
2003/0081582 A1	5/2003	Jain et al.	
2003/0093790 A1	5/2003	Logan et al.	
2003/0103571 A1*	6/2003	Meehan	H04N 21/234327 375/240.27
2003/0107994 A1	6/2003	Jacobs et al.	
2003/0135631 A1	7/2003	Li et al.	
2003/0140159 A1	7/2003	Campbell et al.	
2003/0151753 A1	8/2003	Li et al.	
2003/0152036 A1	8/2003	Quigg Brown et al.	
2003/0154239 A1	8/2003	Davis et al.	
2003/0195977 A1	10/2003	Liu et al.	
2003/0204519 A1	10/2003	Sirivara et al.	
2003/0204602 A1	10/2003	Hudson et al.	
2003/0236904 A1	12/2003	Walpole et al.	
2004/0003101 A1	1/2004	Roth et al.	
2004/0010613 A1	1/2004	Apostolopoulos et al.	
2004/0030547 A1	2/2004	Leaning et al.	
2004/0030599 A1	2/2004	Sie et al.	
2004/0030797 A1	2/2004	Akinlar et al.	
2004/0031054 A1	2/2004	Dankworth et al.	
2004/0049780 A1	3/2004	Gee	
2004/0054551 A1	3/2004	Ausubel et al.	
2004/0071209 A1	4/2004	Burg et al.	
2004/0083283 A1	4/2004	Sundaram et al.	
2004/0093420 A1	5/2004	Gamble	
2004/0103444 A1	5/2004	Weinberg et al.	
2004/0117427 A1	6/2004	Allen et al.	
2004/0143672 A1	7/2004	Padmanabham et al.	
2004/0168052 A1	8/2004	Clisham et al.	
2004/0170392 A1	9/2004	Lu et al.	
2004/0179032 A1	9/2004	Huang	
2004/0199655 A1	10/2004	Davies et al.	
2004/0220926 A1	11/2004	Lamkin et al.	
2004/0221088 A1	11/2004	Lisitsa et al.	
2004/0260701 A1	12/2004	Lehikoinen et al.	
2004/0267956 A1	12/2004	Leon et al.	
2005/0015509 A1	1/2005	Sitaraman	
2005/0033855 A1	2/2005	Moradi et al.	
2005/0055425 A1*	3/2005	Lango	H04L 29/06027 709/219
2005/0066063 A1	3/2005	Grigorovitch et al.	
2005/0076136 A1	4/2005	Cho et al.	
2005/0084166 A1	4/2005	Bonch et al.	
2005/0108414 A1	5/2005	Taylor et al.	
2005/0120107 A1	6/2005	Kagan et al.	
2005/0123058 A1	6/2005	Greenbaum et al.	
2005/0185578 A1	8/2005	Padmanabham et al.	
2005/0188051 A1	8/2005	Sneh	
2005/0204046 A1	9/2005	Watanabe	
2005/0251832 A1	11/2005	Chiueh	
2005/0262257 A1	11/2005	Major et al.	
2006/0010003 A1	1/2006	Kruse	
2006/0059223 A1	3/2006	Klemets et al.	
2006/0075446 A1	4/2006	Klemets et al.	
2006/0080718 A1	4/2006	Gray et al.	
2006/0130118 A1	6/2006	Damm	
2006/0133809 A1	6/2006	Chow et al.	
2006/0165166 A1	7/2006	Chou et al.	
2006/0168290 A1	7/2006	Doron	
2006/1068295	7/2006	Batterberry et al.	
2006/0206246 A1	9/2006	Walker	
2006/0236219 A1	10/2006	Grigorovitch et al.	
2006/0277564 A1	12/2006	Jarman	
2007/0024705 A1	2/2007	Richter et al.	
2007/0030833 A1	2/2007	Pirzada et al.	
2007/0067480 A1	3/2007	Beek et al.	
2007/0079325 A1	4/2007	de Heer	
2007/0094405 A1	4/2007	Zhang	
2007/0204310 A1	8/2007	Hua et al.	
2007/0280255 A1	12/2007	Tsang et al.	
2008/0028428 A1	1/2008	Jeong et al.	
2008/0037527 A1	2/2008	Chan et al.	
2008/0046939 A1	2/2008	Lu et al.	
2008/0056373 A1	3/2008	Newlin et al.	
2008/0104647 A1	5/2008	Hannuksela	
2008/0120330 A1	5/2008	Reed et al.	
2008/0120342 A1	5/2008	Reed et al.	
2008/0133766 A1	6/2008	Luo	
2008/0162713 A1	7/2008	Bowra et al.	
2008/0184688 A1	8/2008	Daly et al.	
2008/0195744 A1	8/2008	Bowra et al.	
2008/0205291 A1	8/2008	Li et al.	
2008/0219151 A1	9/2008	Ma et al.	
2008/0222235 A1	9/2008	Hurst et al.	
2008/0263180 A1	10/2008	Hurst et al.	
2008/0281803 A1	11/2008	Gentric	
2009/0043906 A1	2/2009	Hurst et al.	
2009/0055471 A1	2/2009	Kozat et al.	
2009/0055547 A1	2/2009	Hudson et al.	
2009/0210549 A1	8/2009	Hudson et al.	
2010/0098103 A1	4/2010	Xiong et al.	
2010/0262711 A1	10/2010	Bouazizi	
2011/0307545 A1	12/2011	Bouazizi	
2015/0058496 A1	2/2015	Hurst et al.	

FOREIGN PATENT DOCUMENTS

EP	1202487 A2	5/2002
EP	1298931 A2	4/2003
EP	1395014 A1	3/2004
EP	1670256 A2	6/2006
EP	1777969	4/2007
GB	2367219 A	3/2002
JP	2000-201343	7/2000
JP	200192752	4/2001
JP	2011004225 A	1/2011
WO	2001067264 A1	9/2001
WO	2004025405 A2	3/2004
WO	2006010113 A2	1/2006

OTHER PUBLICATIONS

Bommaiah, E., et al., "Design and Implementation of a Caching System for Streaming Media over the Internet," 2000 IEEE. Published in RTAS '00 Proceedings of the Sixth IEEE Real Time Technology and Applications Symposium (RTAS 2000), p. 111.

Krasic et al., Quality-Adaptive Media Streaming by Priority Drop, Oregon Graduate Institute, 2001.

Krasic et al., QoS Scalability for Streamed Media Delivery, Oregon Graduate Institute School of Science & Engineering Technical Report CSE 99-011, Sep. 1999.

Huang et al., Adaptive Live Video Streaming by Priority Drop, Portland State University PDXScholar, Jul. 21, 2003.

(56) **References Cited**

OTHER PUBLICATIONS

Walpole et al, A Player for Adapctive MPEG Video Streaming Over the Internet, Oregon Graduate Institute of Science and Technology, Oct. 25, 2012.
Albanese, Andrew et al. "Priority Encoding Transmission", TR-94-039, Aug. 1994, 36 pgs, International Computer Science Institute, Berkeley, CA.
Birney, Bill "Intelligent Streaming", May 2003, Microsoft.
Goyal, Vivek K. "Multiple Description Coding: Compression Meets the Network," Sep. 2001, pp. 74-93, IEEE Signal Processing Magazine.
ON2 Technologies, Inc. "TrueMotion VP7 Video Codec" White Paper, Document Version 1.0, Jan. 10, 2005.
Pathan, Al-Mukaddim et al. "A Taxonomy and Survey of Content Delivery Networks" Australia, Feb. 2007, available at http://www.gridbus.org/reports/CDN-Taxonomy.pdf.
Puri, Rohit et al. "Multiple Description Source Coding Using Forward Error Correction Codes," Oct. 1999, 5 pgs., Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA.
Wicker, Stephen B. "Error Control Systems for Digital Communication and Storage," Prentice-Hall, Inc., New Jersey, USA, 1995, parts 1-6.
Liu, Jiangchuan et al. "Opportunities and Challenged of Peer-to-Peer Internet Video Broadcast," School of Computing Science, Simon Fraser University, British Columbia, Canada.
Clement, B. "Move Networks closes $11.3 Million on First Round VC Funding," Page One PR, Move Networks, Inc. Press Releases, Feb. 7, 2007, http://www.move.tv/press/press20070201.html.
Move Networks, Inc. "The Next Generation Video Publishing System," Apr. 11, 2007; http://www.movenetworks.com/wp-content/uploads/move-networks-publishing-system.pdf.
Yoshimura, Takeshi et al. "Mobile Streaming Media CDN Enabled by Dynamic SMIL", NTT DoCoMo, Multimedia Laboratories and Hewlett-Packard Laboratories,dated May 7-11, 2002, ACM 1-58113-449-5/02/0005; http://www2002.org/CDROM/refereed/515/.
Nguyen, T. et al., Multiple Sender Distributed Video Streaming, IEEE Transactinos on Multimedia, IEEE Service Center, Piscataway, NJ, US, vol. 6, No. 2, Apr. 1, 2004, pp. 315-326, XP011109142, ISSN: 1520-9210, DOI: 10.1109/TMM,2003.822790.
Fujisawa, Hiroshi et al. "Implementaton of Efficient Access Mechanism for Multiple Mirror-Servers" IPSJ SIG Technical Report, vol. 2004, No. 9 (2004-DPS-116), Jan. 30, 2004, Information Processing Society of Japan, pp. 37-42.
Liu, Jiangchuan et al. "Adaptive Video Multicast Over the Internet" IEEE Computer Society, 2003.
"The meaning of performance factor—English-Japanese Weblio Dictionary", [online], Feb. 24, 2012, [searched on Feb. 24, 2012], the Internet <URL:http://ejje.weblio.jp/content/performance+factor<.
Tsuru, et al. "Recent evolution of the Internet measurement and inference techniques", IEICE Technical Report, vol. 103, No. 123, pp. 37-42, Jun. 12, 2003.
Rejaie, Reza et al. "Architectural Considerations for Playback of Quality Adaptive Video OVer the Internet" University of Southern California, Information Sciences Institute, 1998.
Roy, Sumit et al. "A System Architecture for Managing Mobile Streaming Media Services" Streaming Media Systems Group, Hewlett-Packard Laboratories, 2003.
Xu, Dongyan et al. "On Peer-to-Peer Media Streaming" Department of Computer Sciences, Purdue University.
Kozamerink, Franc "Media Streaming Over the Internet—An Over of Delivery Technologies" EBU Technical Review, Oct. 2002.
Lienhart, Rainer et al. "Challenges in Distributed Video Management and Delivery" Intel Corporation, EECS Dept., UC Berkeley, 2000-2002.
Zhang, Xinyan et al. "CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming" IEEE 2005.
Guo, Yang "DirectStream: A Directory-Based Peer-to-Peer Video Streaming Service" LexisNexis, Elsevier B.V. 2007.

* cited by examiner

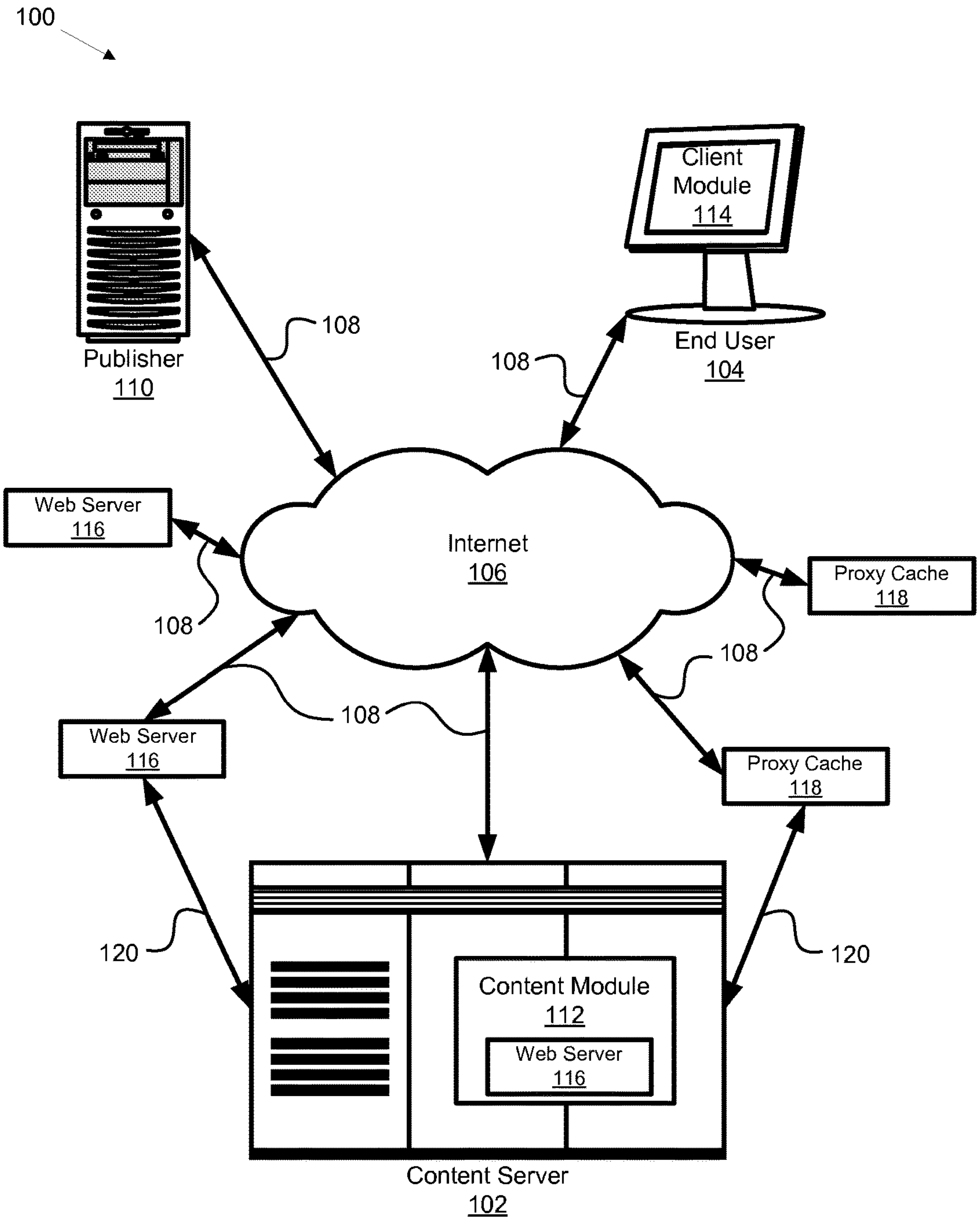

100
Publisher
110
Client
Module
114
End User
104
108
108
Web Server
116
Internet
106
Proxy Cache
118
108
108
108
Web Server
116
Proxy Cache
118
120
Content Module
112
Web Server
116
120
Content Server
102

FIG. 1

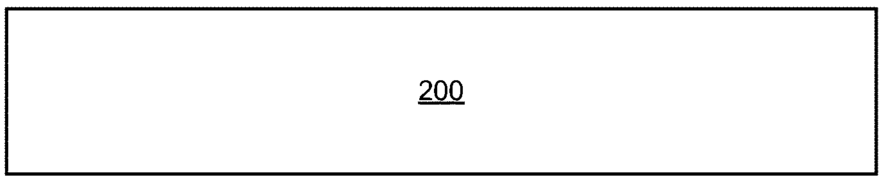


FIG. 2a

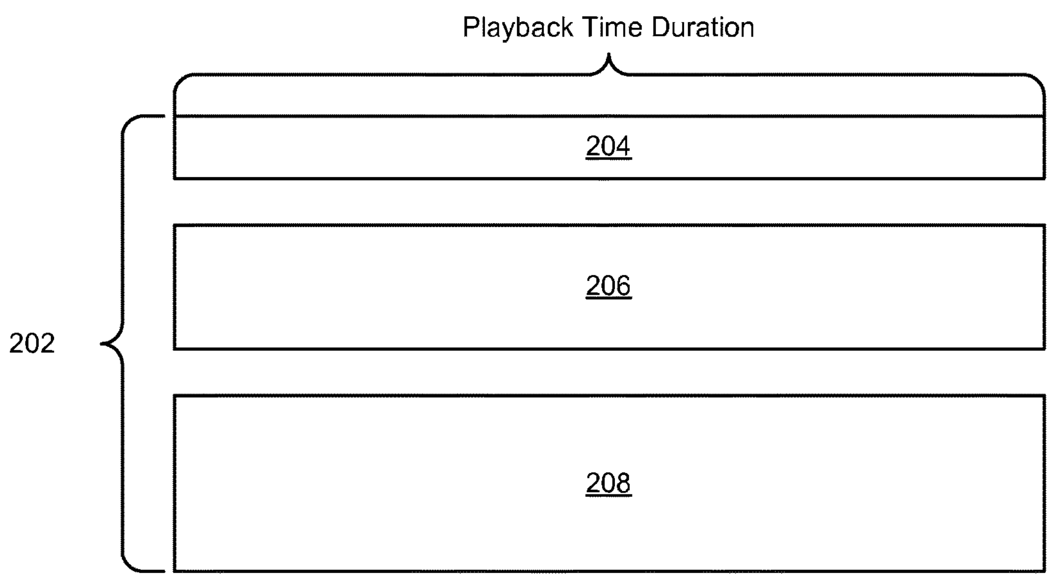


FIG. 2b

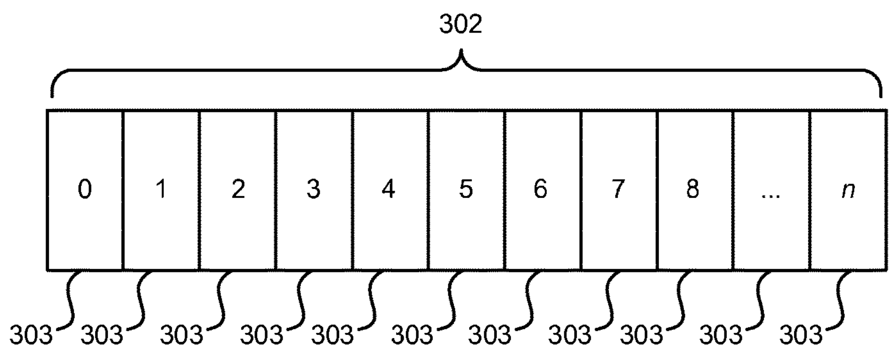

302
0
1
2
3
4
5
6
7
8
...
n
303

FIG. 3a

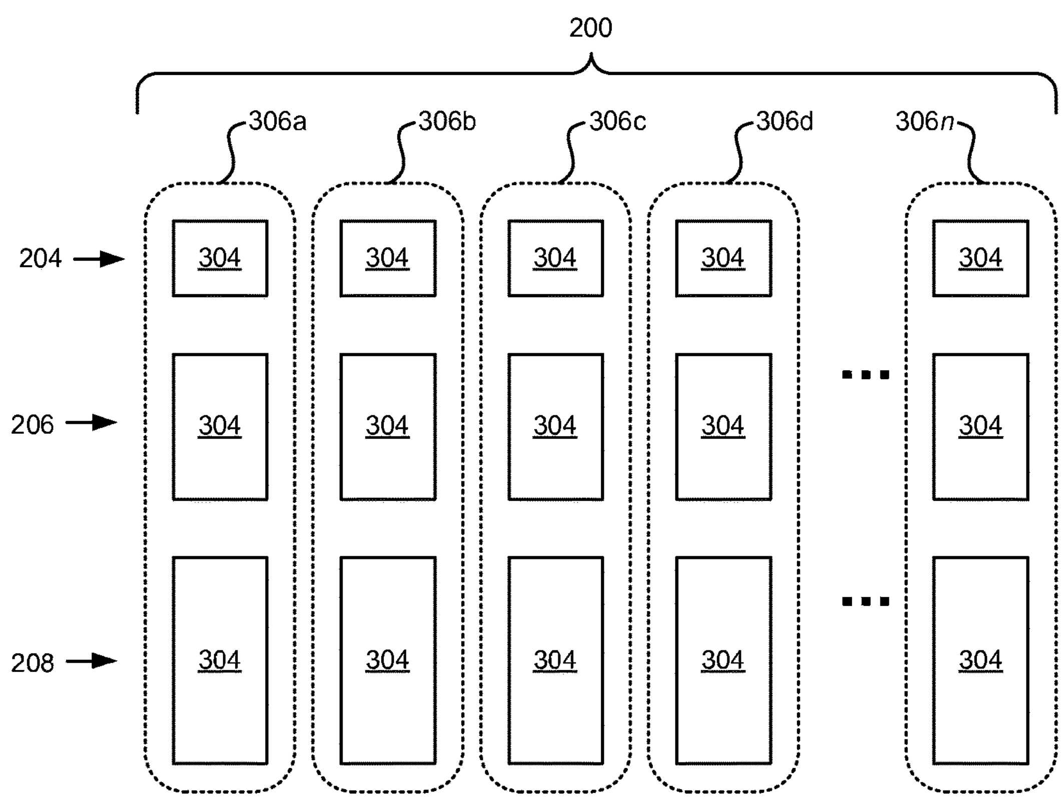

200
306a
306b
306c
306d
306n
204
206
208
304

FIG. 3b

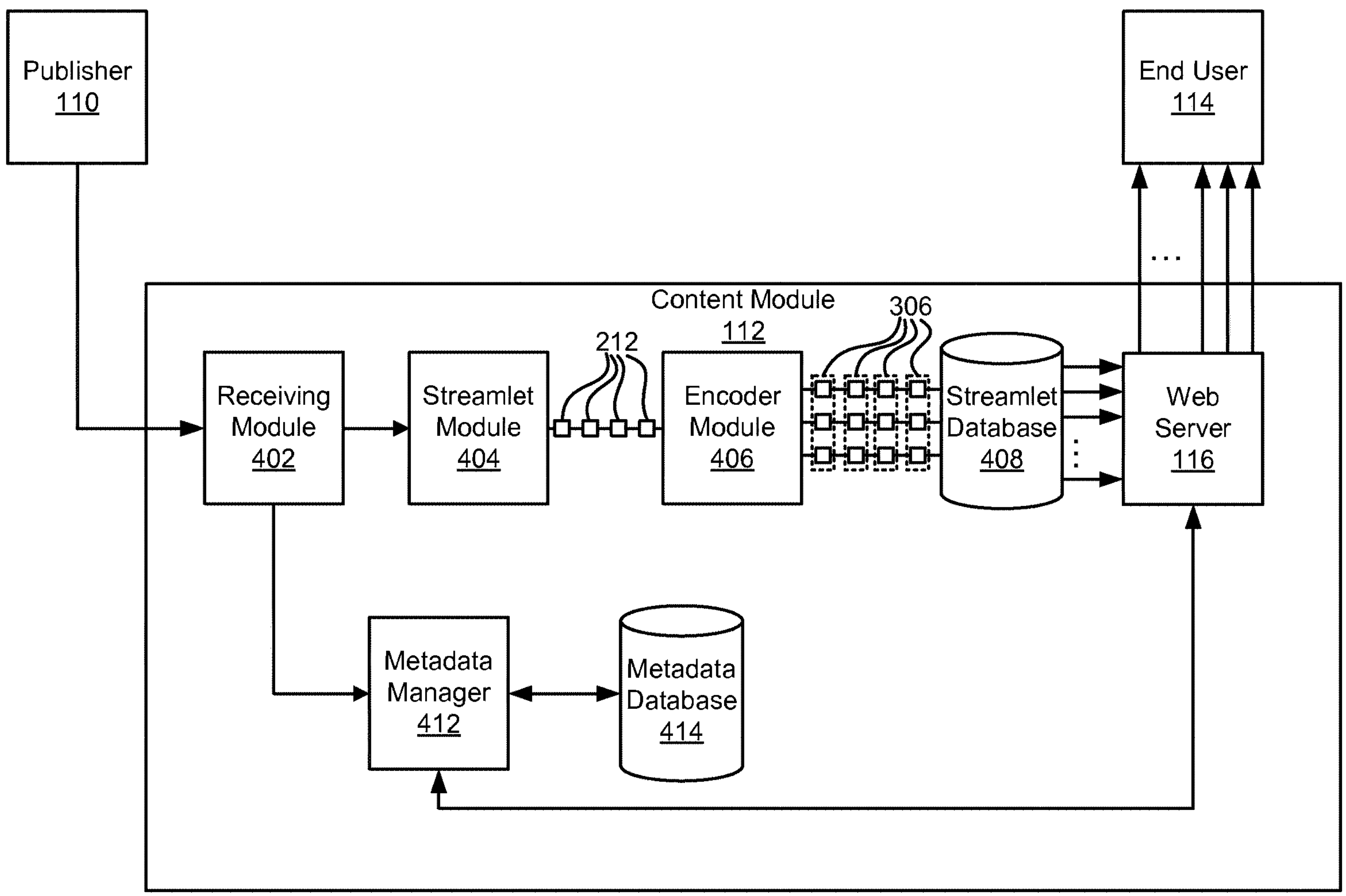

Publisher
110
End User
114
Content Module
112
306
212
Receiving
Module
402
Streamlet
Module
404
Encoder
Module
406
Streamlet
Database
408
Web
Server
116
Metadata
Manager
412
Metadata
Database
414

FIG. 4

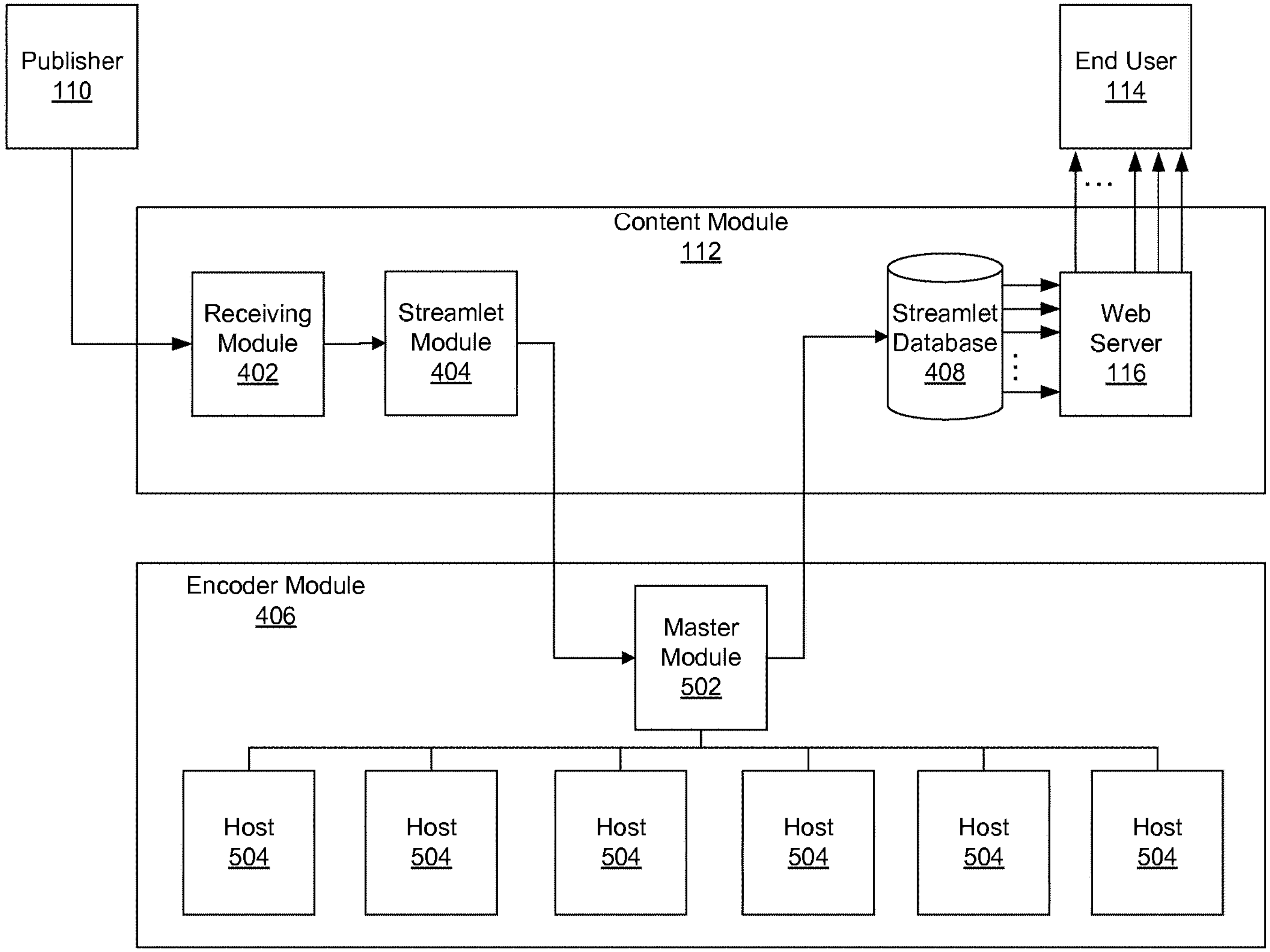

Publisher
110
End User
114
Content Module
112
Receiving Module
402
Streamlet Module
404
Streamlet Database
408
Web Server
116
Encoder Module
406
Master Module
502
Host
504
Host
504
Host
504
Host
504
Host
504
Host
504

FIG. 5a

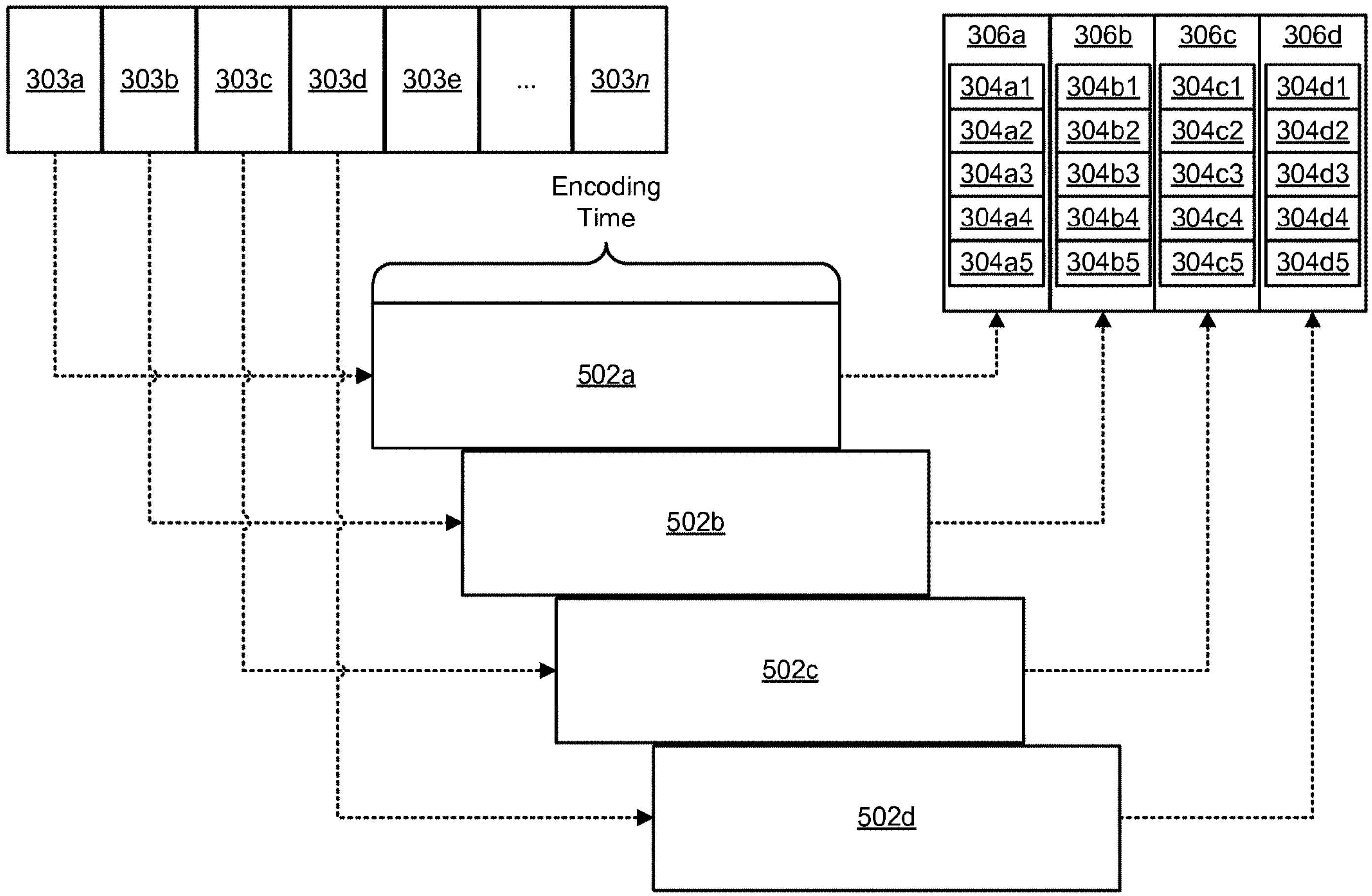
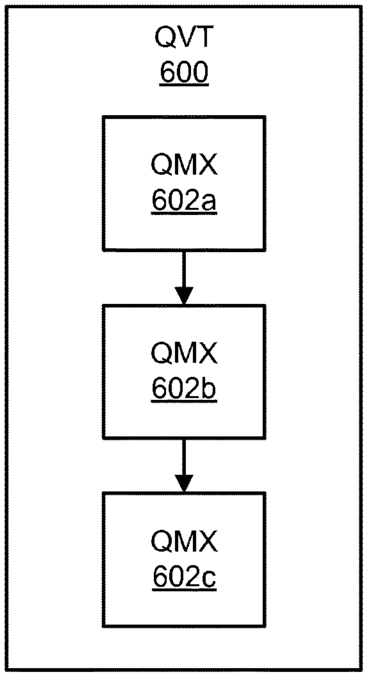

303a
303b
303c
303d
303e
...
303n
Encoding Time
502a
502b
502c
502d
306a
306b
306c
306d
304a1
304a2
304a3
304a4
304a5
304b1
304b2
304b3
304b4
304b5
304c1
304c2
304c3
304c4
304c5
304d1
304d2
304d3
304d4
304d5

FIG. 5b



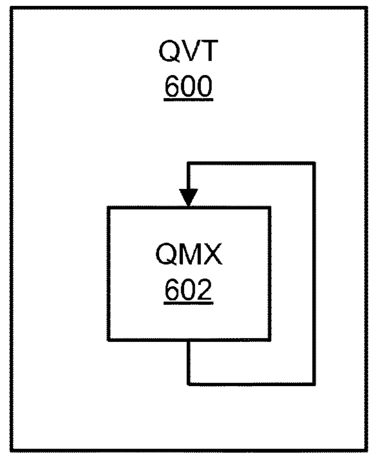


FIG. 6b

FIG. 6a

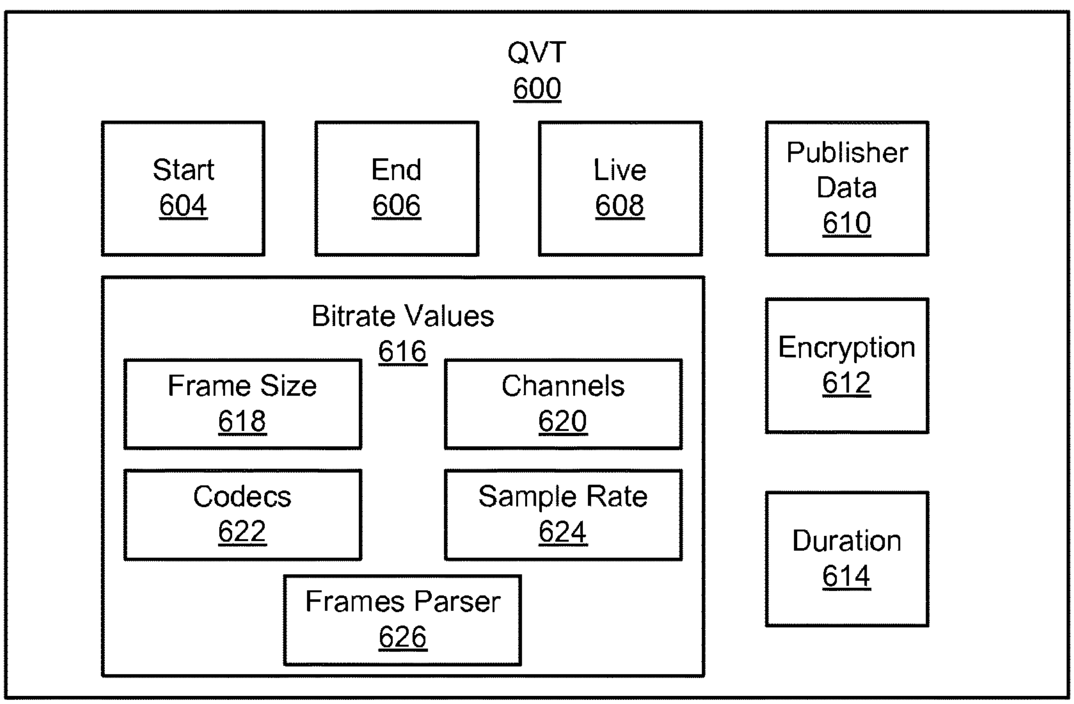


FIG. 6c

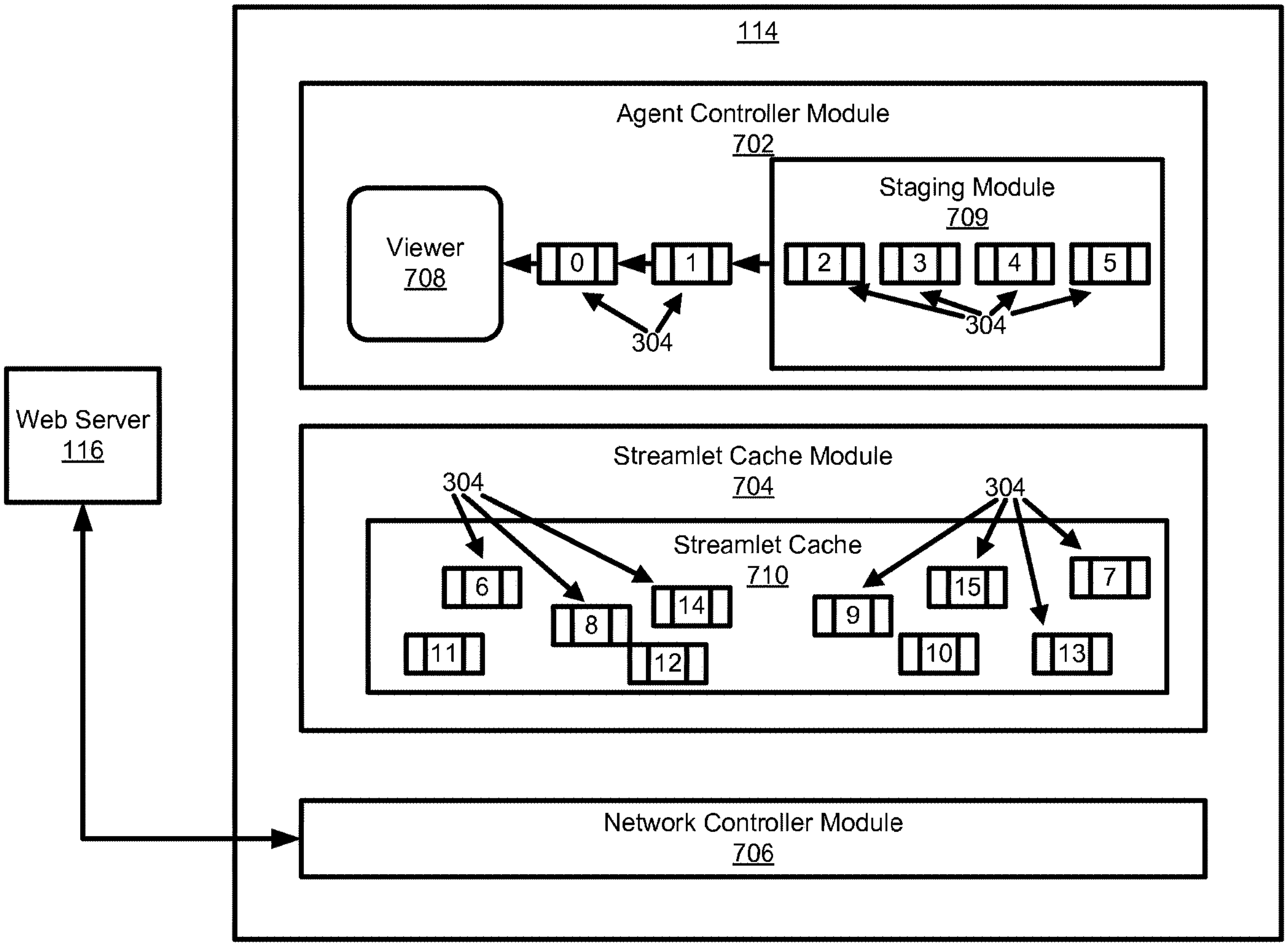

114
Agent Controller Module
702
Viewer
708
0
1
304
Staging Module
709
2
3
4
5
304
Web Server
116
Streamlet Cache Module
704
304
Streamlet Cache
710
304
6
8
14
11
12
9
15
7
10
13
Network Controller Module
706

FIG. 7

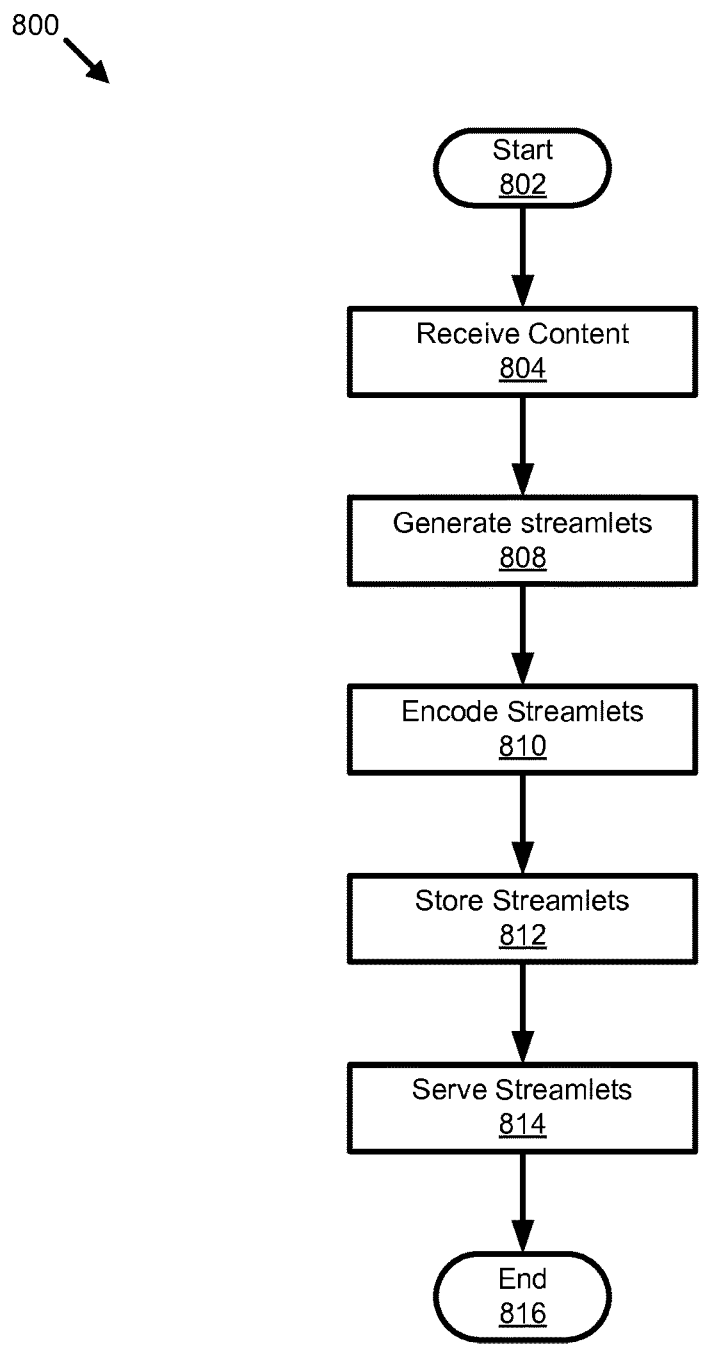
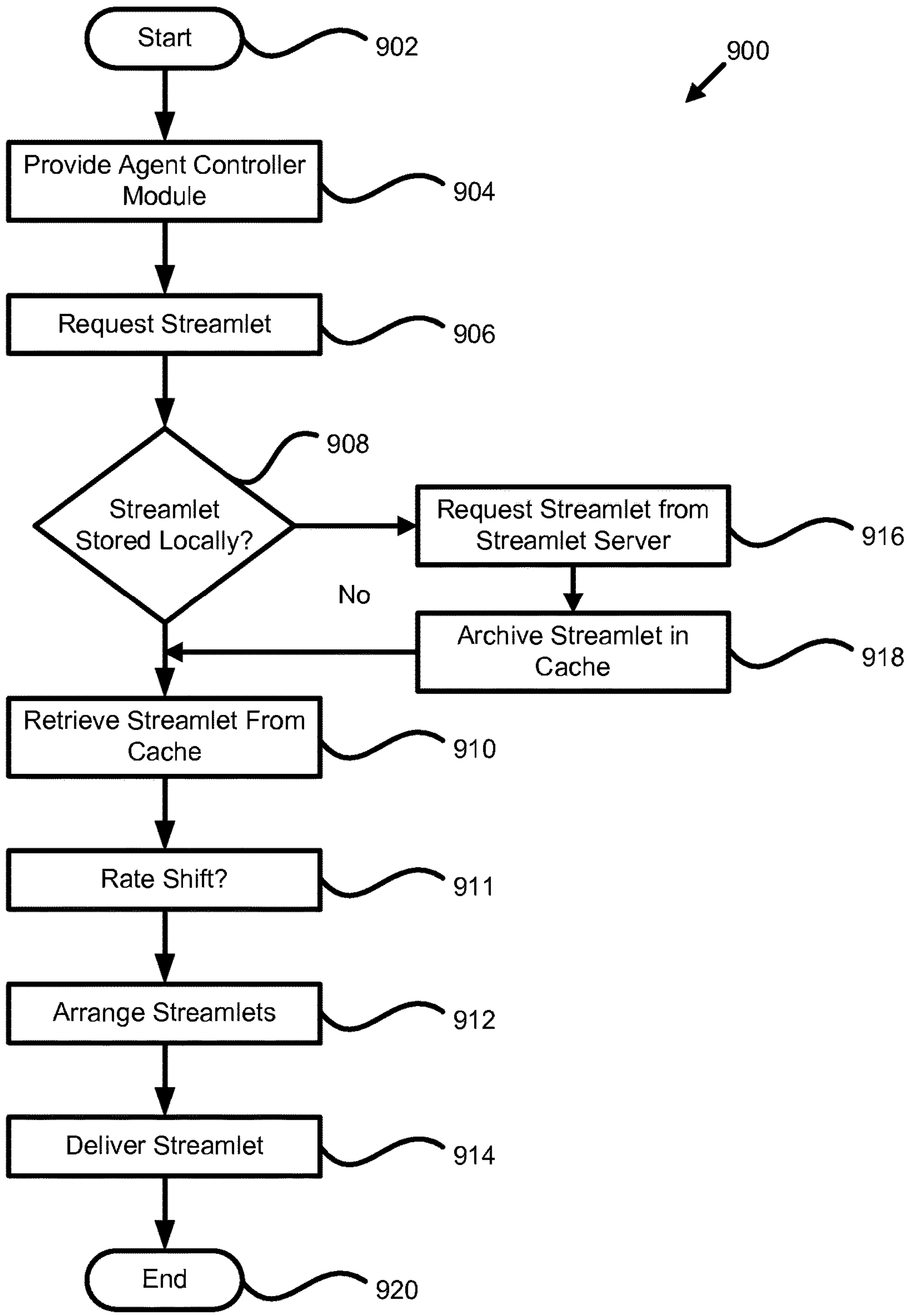
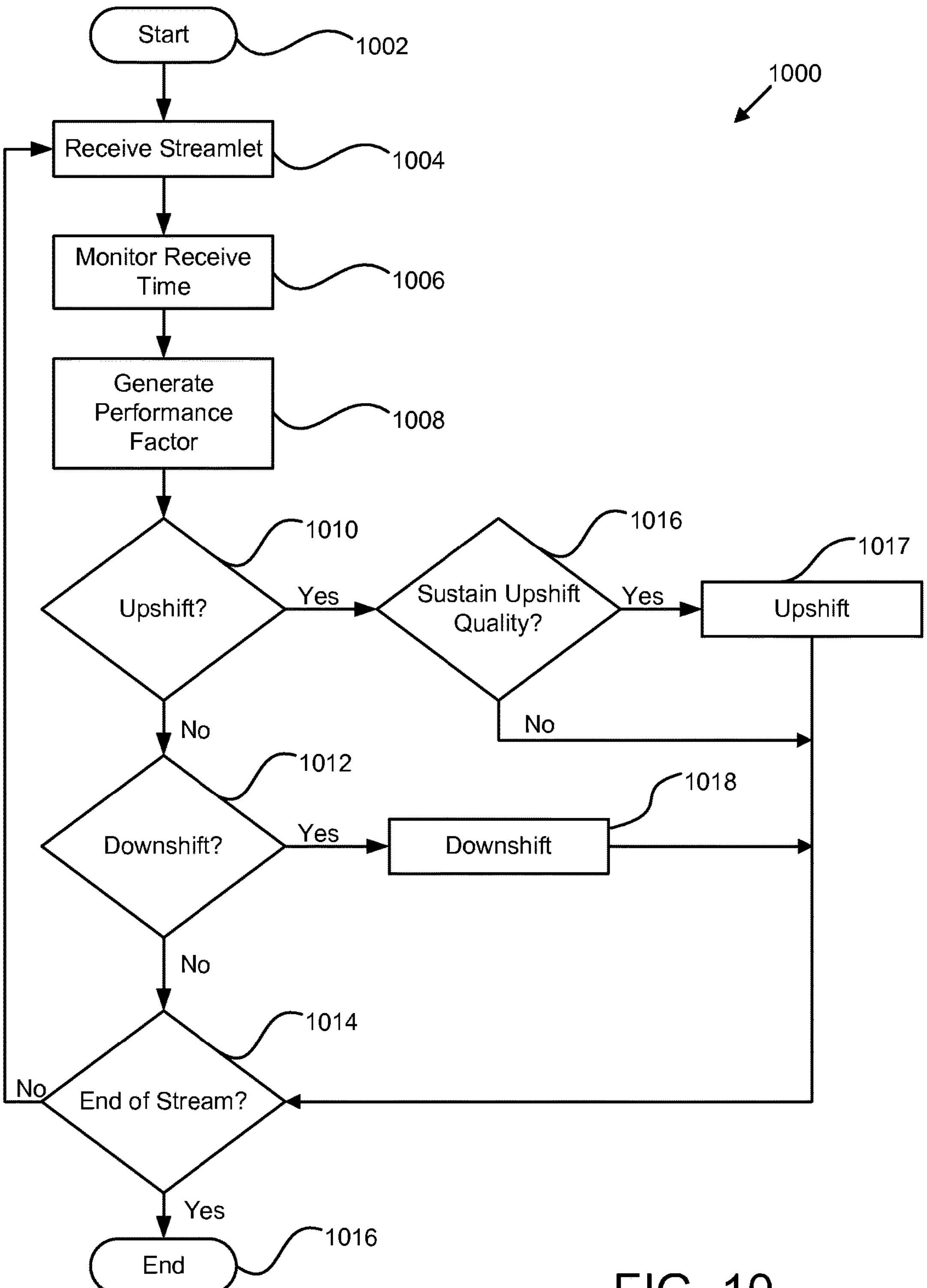

800
Start
802
Receive Content
804
Generate streamlets
808
Encode Streamlets
810
Store Streamlets
812
Serve Streamlets
814
End
816

FIG. 8



Start
902
900
Provide Agent Controller Module
904
Request Streamlet
906
908
Streamlet Stored Locally?
Request Streamlet from Streamlet Server
916
No
Archive Streamlet in Cache
918
Retrieve Streamlet From Cache
910
Rate Shift?
911
Arrange Streamlets
912
Deliver Streamlet
914
End
920

FIG. 9



Start
1002
1000
Receive Streamlet
1004
Monitor Receive Time
1006
Generate Performance Factor
1008
1010
Upshift?
Yes
1016
Sustain Upshift Quality?
Yes
1017
Upshift
No
No
1012
Downshift?
Yes
1018
Downshift
No
1014
No
End of Stream?
Yes
1016
End

FIG. 10

APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 16/004,056 filed on Jun. 8, 2018, which is a continuation of U.S. patent application Ser. No. 15/414,027 (now U.S. Pat. No. 9,998,516) filed on Jan. 24, 2017, which is a continuation of U.S. patent application Ser. No. 14/719,122 filed on May 21, 2015, which is a continuation of U.S. patent application Ser. No. 14/106,051 filed on Dec. 13, 2013 (now U.S. Pat. No. 9,071,668), which is a continuation of U.S. patent application Ser. No. 13/617,114, filed on Sep. 14, 2012 (now U.S. Pat. No. 8,612,624), which is a continuation of U.S. patent Ser. No. 12/906,940 filed on Oct. 18, 2010 (now U.S. Pat. No. 8,402,156), which is a continuation of U.S. patent application Ser. No. 11/673,483, filed on Feb. 9, 2007 (now U.S. Pat. No. 7,818,444), which is a continuation-in-part of application Ser. No. 11/116,783, filed on Apr. 28, 2005 (now U.S. Pat. No. 8,868,772), which claims the benefit of U.S. Provisional Application No. 60/566,831, filed on Apr. 31, 2004, all of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to video streaming over packet switched networks such as the Internet, and more particularly relates to adaptive-rate shifting of streaming content over such networks.

Description of the Related Art

The Internet is fast becoming a preferred method for distributing media files to end users. It is currently possible to download music or video to computers, cell phones, or practically any network capable device. Many portable media players are equipped with network connections and enabled to play music or videos. The music or video files (hereinafter "media files") can be stored locally on the media player or computer, or streamed or downloaded from a server.

"Streaming media" refers to technology that delivers content at a rate sufficient for presenting the media to a user in real time as the data is received. The data may be stored in memory temporarily until played and then subsequently deleted. The user has the immediate satisfaction of viewing the requested content without waiting for the media file to completely download. Unfortunately, the audio/video quality that can be received for real time presentation is constrained by the available bandwidth of the user's network connection. Streaming may be used to deliver content on demand (previously recorded) or from live broadcasts.

Alternatively, media files may be downloaded and stored on persistent storage devices, such as hard drives or optical storage, for later presentation. Downloading complete media files can take large amounts of time depending on the network connection. Once downloaded, however, the content can be viewed repeatedly anytime or anywhere. Media files prepared for downloading usually are encoded with a higher quality audio/video than can be delivered in real time. Users generally dislike this option, as they tend to want to see or hear the media file instantaneously.

Streaming offers the advantage of immediate access to the content but currently sacrifices quality compared with downloading a file of the same content. Streaming also provides the opportunity for a user to select different content for viewing on an ad hoc basis, while downloading is by definition restricted to receiving a specific content selection in its entirety or not at all. Downloading also supports rewind, fast forward, and direct seek operations, while streaming is unable to fully support these functions. Streaming is also vulnerable to network failures or congestion.

Another technology, known as "progressive downloads," attempts to combine the strengths of the above two technologies. When a progressive download is initiated, the media file download begins, and the media player waits to begin playback until there is enough of the file downloaded that playback can begin with the hope that the remainder of the file will be completely downloaded before playback "catches up." This waiting period before playback can be substantial depending on network conditions, and therefore is not a complete or fully acceptable solution to the problem of media presentation over a network.

Generally, three basic challenges exist with regard to data transport streaming over a network such as the Internet that has a varying amount of data loss. The first challenge is reliability. Most streaming solutions use a TCP connection, or "virtual circuit," for transmitting data. A TCP connection provides a guaranteed delivery mechanism so that data sent from one endpoint will be delivered to the destination, even if portions are lost and retransmitted. A break in the continuity of a TCP connection can have serious consequences when the data must be delivered in real-time. When a network adapter detects delays or losses in a TCP connection, the adapter "backs off" from transmission attempts for a moment and then slowly resumes the original transmission pace. This behavior is an attempt to alleviate the perceived congestion. Such a slowdown is detrimental to the viewing or listening experience of the user and therefore is not acceptable.

The second challenge to data transport is efficiency. Efficiency refers to how well the user's available bandwidth is used for delivery of the content stream. This measure is directly related to the reliability of the TCP connection. When the TCP connection is suffering reliability problems, a loss of bandwidth utilization results. The measure of efficiency sometimes varies suddenly, and can greatly impact the viewing experience.

The third challenge is latency. Latency is the time measure form the client's point-of-view, of the interval between when a request is issued and the response data begins to arrive. This value is affected by the network connection's reliability and efficiency, and the processing time required by the origin to prepare the response. A busy or overloaded server, for example, will take more time to process a request. As well as affecting the start time of a particular request, latency has a significant impact on the network throughput of TCP.

From the foregoing discussion, it should be apparent that a need exists for an apparatus, system, and method that alleviate the problems of reliability, efficiency, and latency. Additionally, such an apparatus, system, and method would offer instantaneous viewing along with the ability to fast forward, rewind, direct seek, and browse multiple streams. Beneficially, such an apparatus, system, and method would utilize multiple connections between a source and destination, requesting varying bitrate streams depending upon network conditions.

SUMMARY OF THE INVENTION

The present invention has been developed in response to the present state of the art, and in particular, in response to the problems and needs in the art that have not yet been fully solved by currently available content streaming systems. Accordingly, the present invention has been developed to provide an apparatus, system, and method for adaptive-rate content streaming that overcome many or all of the above-discussed shortcomings in the art.

The apparatus for adaptive-rate content streaming is provided with a logic unit containing a plurality of modules configured to functionally execute the necessary steps. These modules in the described embodiments include a receiving module configured to receive media content, a streamlet module configured to segment the media content and generate a plurality of sequential streamlets, and an encoding module configured to encode each streamlet as a separate content file.

The encoding module is further configured to generate a set of streamlets for each of the sequential streamlets. Each streamlet may comprise a portion of the media content having a predetermined length of time. The predetermined length of time may be in the range of between about 0.1 and 5 seconds.

In one embodiment, a set of streamlets comprises a plurality of streamlets having identical time indices, and each streamlet of the set of streamlets has a unique bitrate. The receiving module is configured to convert the media content to raw audio or raw video. The encoding module may include a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid. The job completion bid may be based on a plurality of computing variables selected from a group consisting of current encoding job completion percentage, average encoding job completion time, processor speed, and physical memory capacity.

A system of the present invention is also presented for adaptive-rate content streaming. In particular, the system, in one embodiment, includes a receiving module configured to receive media content, a streamlet module configured to segment the media content and generate a plurality of sequential streamlets, each streamlet comprising a portion of the media content having a predetermined length of time, and an encoding module configured to encode each streamlet as a separate content file and generate a set of streamlets.

The system also includes a plurality of streamlets having identical time indices and each streamlet of the set of streamlets having a unique bitrate. The encoding module comprises a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid.

A method of the present invention is also presented for adaptive-rate content streaming. In one embodiment, the method includes receiving media content, segmenting the media content and generating a plurality of sequential streamlets, and encoding each streamlet as a separate content file.

The method also includes segmenting the media content into a plurality of streamlets, each streamlet comprising a portion of the media content having a predetermined length of time. In one embodiment, the method includes generating a set of streamlets comprising a plurality of streamlets having identical time indices, and each streamlet of the set of streamlets having a unique bitrate.

Furthermore, the method may include converting the media content to raw audio or raw video, and segmenting the content media into a plurality of sequential streamlets. The method further comprises assigning an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid, and submitting an encoding job completion bid based on a plurality of computing variables.

Reference throughout this specification to features, advantages, or similar language does not imply that all of the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussion of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

Furthermore, the described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize that the invention may be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

These features and advantages of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the advantages of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings, in which:

FIG. **1** is a schematic block diagram illustrating one embodiment of a system for dynamic rate shifting of streaming content in accordance with the present invention;

FIG. **2***a* is a schematic block diagram graphically illustrating one embodiment of a media content file;

FIG. **2***b* is a schematic block diagram illustrating one embodiment of a plurality of streams having varying degrees of quality and bandwidth;

FIG. **3***a* is a schematic block diagram illustrating one embodiment of a stream divided into a plurality of source streamlets;

FIG. **3***b* is a schematic block diagram illustrating one embodiment of sets of streamlets in accordance with the present invention;

FIG. **4** is a schematic block diagram illustrating in greater detail one embodiment of the content module in accordance with the present invention;

FIG. **5***a* is a schematic block diagram illustrating one embodiment of an encoder module in accordance with the present invention;

FIG. **5***b* is a schematic block diagram illustrating one embodiment of parallel encoding of streamlets in accordance with the present invention;

FIG. **6***a* is a schematic block diagram illustrating one embodiment of a virtual timeline in accordance with the present invention;

FIG. **6***b* is a schematic block diagram illustrating an alternative embodiment of a VT in accordance with the present invention;

FIG. **6***c* is a schematic block diagram illustrating one embodiment of a QMX in accordance with the present invention;

FIG. **7** is a schematic block diagram graphically illustrating one embodiment of a client module in accordance with the present invention;

FIG. **8** is a schematic flow chart diagram illustrating one embodiment of a method for processing content in accordance with the present invention;

FIG. **9** is a schematic flow chart diagram illustrating one embodiment of a method for viewing a plurality of streamlets in accordance with the present invention; and

FIG. **10** is a schematic flow chart diagram illustrating one embodiment of a method for requesting streamlets within an adaptive-rate shifting content streaming environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Many of the functional units described in this specification have been labeled as modules, in order to more particularly emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom VLSI circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices or the like.

Modules may also be implemented in software for execution by various types of processors. An identified module of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions which may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may comprise disparate instructions stored in different locations which, when joined logically together, comprise the module and achieve the stated purpose for the module.

Indeed, a module of executable code may be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

Reference throughout this specification to "one embodiment," "an embodiment," or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in one embodiment," "in an embodiment," and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

Reference to a signal bearing medium may take any form capable of generating a signal, causing a signal to be generated, or causing execution of a program of machine-readable instructions on a digital processing apparatus. A signal bearing medium may be embodied by a transmission line, a compact disk, digital-video disk, a magnetic tape, a Bernoulli drive, a magnetic disk, a punch card, flash memory, integrated circuits, or other digital processing apparatus memory device. In one embodiment, a computer program product including a computer useable medium having a computer readable program of computer instructions stored thereon that when executed on a computer causes the computer to carry out operations for multi-bitrate content streaming as described herein.

Furthermore, the described features, structures, or characteristics of the invention may be combined in any suitable manner in one or more embodiments. In the following description, numerous specific details are provided, such as examples of programming, software modules, user selections, network transactions, database queries, database structures, hardware modules, hardware circuits, hardware chips, etc., to provide a thorough understanding of embodiments of the invention. One skilled in the relevant art will recognize, however, that the invention may be practiced without one or more of the specific details, or with other methods, components, materials, and so forth. In other instances, well-known structures, materials, or operations are not shown or described in detail to avoid obscuring aspects of the invention.

FIG. **1** is a schematic block diagram illustrating one embodiment of a system **100** for dynamic rate shifting of streaming content in accordance with the present invention. In one embodiment, the system **100** comprises a content server **102** and an end user station **104**. The content server **102** and the end user station **104** may be coupled by a data communications network. The data communications network may include the Internet **106** and connections **108** to the Internet **106**. Alternatively, the content server **102** and the end user **104** may be located on a common local area network, wireless area network, cellular network, virtual local area network, or the like. The end user station. **104** may comprise a personal computer (PC), an entertainment system configured to communicate over a network, or a portable electronic device configured to present content. For example, portable electronic devices may include, but are not limited to, cellular phones, portable gaming systems, and portable computing devices.

In the depicted embodiment, the system **100** also includes a publisher **110**, and a web server **116**. The publisher **110** may be a creator or distributor of content. For example, if the content to be streamed were a broadcast of a television program, the publisher **110** may be a television or cable network channel such as NBC®, or MTV®. Content may be transferred over the Internet **106** to the content server **102**, where the content is received by a content module **112**. The content module **112** may be configured to receive, process, and store content. In one embodiment, processed content is accessed by a client module **114** configured to play the content on the end user station **104**. In a further embodiment, the client module **114** is configured to receive different portions of a content stream from a plurality of locations simultaneously. For example, the client module **114** may request and receive content from any of the plurality of web servers **116**.

Content from the content server **102** may be replicated to other web servers **116** or alternatively to proxy cache servers **118**. Replicating may occur by deliberate forwarding from the content server **102**, or by a web, cache, or proxy server outside of the content server **102** asking for content on behalf of the client module **114**. In a further embodiment, content may be forwarded directly to web **116** or proxy **118** servers through direct communication channels **120** without the need to traverse the Internet **106**.

FIG. **2***a* is a schematic block diagram graphically illustrating one embodiment of a media content (hereinafter "content") file **200**. In one embodiment, the content file **200** is distributed by the publisher **110**. The content file **200** may comprise a television broadcast, sports event, movie, music, concert, etc. The content file **200** may also be live or archived content. The content file **200** may comprise uncompressed video and audio, or alternatively, video or audio. Alternatively, the content file **200** may be compressed using standard or proprietary encoding schemes. Examples of encoding schemes capable of use with the present invention include, but are not limited to, DivX®, Windows Media Video®, Quicktime Sorenson 3®, On2, OGG Vorbis, MP3, or Quicktime 6.5/MPEG-4® encoded content.

FIG. **2***b* is a schematic block diagram illustrating one embodiment of a plurality of streams **202** having varying degrees of quality and bandwidth. In one embodiment, the plurality of streams **202** comprises a low quality stream **204**, a medium quality stream **206**, and a high quality stream **208**. Each of the streams **204**, **206**, **208** is a copy of the content file **200** encoded and compressed to varying bit rates. For example, the low quality stream **204** may be encoded and compressed to a bit rate of 100 kilobits per second (kbps), the medium quality stream **206** may be encoded and compressed to a bit rate of 200 kbps, and the high quality stream **208** may be encoded and compressed to 600 kbps.

FIG. **3***a* is a schematic block diagram illustrating one embodiment of a stream **302** divided into a plurality of source streamlets **303**. As used herein, streamlet refers to any sized portion of the content file **200**. Each streamlet **303** may comprise a portion of the content contained in stream **302**, encapsulated as an independent media object. The content in a streamlet **303** may have a unique time index in relation to the beginning of the content contained in stream **302**. In one embodiment, the content contained in each streamlet **303** may have a duration of two seconds. For example, streamlet **0** may have a time index of 00:00 representing the beginning of content playback, and streamlet **1** may have a time index of 00:02, and so on. Alternatively, the time duration of the streamlets **304** may be any duration smaller than the entire playback duration of the content in stream **302**. In a further embodiment, the streamlets **303** may be divided according to file size instead of a time index and duration.

FIG. **3***b* is a schematic block diagram illustrating one embodiment of sets **306** of streamlets in accordance with the present invention. As used herein, the term "set" refers to a group of streamlets having identical time indices and durations but varying bitrates. In the depicted embodiment, the set **306***a* encompasses all streamlets having a time index of 00:00. The set **306***a* includes encoded streamlets **304** having low, medium, and high 204, 206, 208 bitrates. Of course each set **306** may include more than the depicted three bitrates which are given by way of example only. One skilled in the art will recognize that any number of streams having different bitrates may be generated from the original content **200**.

As described above, the duration of one streamlet **304** may be approximately two seconds. Likewise each set **306** may comprise a plurality of streamlets **304** where each streamlet **304** has a playable duration of two seconds. Alternatively, the duration of the streamlet **304** may be predetermined or dynamically variable depending upon a variety of factors including, but not limited to, network congestion, system specifications, playback resolution and quality, etc. In the depicted embodiment, the content **200** may be formed of the plurality of sets **306**. The number of sets **306** may depend on the length of the content **200** and the length or duration of each streamlet **304**.

FIG. **4** is a schematic block diagram illustrating in greater detail one embodiment of the content module **112** in accordance with the present invention. The content module **112** may comprise a capture module **402**, a streamlet module **404**, an encoder module **406**, a streamlet database **408**, and the web server **116**. In one embodiment, the capture module **402** is configured to receive the content file **200** from the publisher **110**. The capture module **402** may be configured to "decompress" the content file **200**. For example, if the content file **200** arrives having been encoded with one of the above described encoding schemes, the capture module **402** may convert the content file **200** into raw audio and/or video. Alternatively, the content file **200** may be transmitted by the publisher in a format **110** that does not require decompression.

The capture module **402** may comprise a capture card configured for TV and/or video capture. One example of a capture card suitable for use in the present invention is the DRC-2500 by Digital Rapids of Ontario, Canada. Alternatively, any capture card capable of capturing audio and video may be utilized with the present invention. In a further embodiment, the capture module **402** is configured to pass the content file to the streamlet module **404**.

The streamlet module **404**, in one embodiment, is configured to segment the content file **200** and generate source streamlets **303** that are not encoded. As used herein, the term "segment" refers to an operation to generate a streamlet of the content file **200** having a duration or size equal to or less than the duration or size of the content file **200**. The streamlet module **404** may be configured to segment the content file **200** into streamlets **303** each having an equal duration. Alternatively, the streamlet module **404** may be configured to segment the content file **200** into streamlets **303** having equal file sizes.

The encoding module **406** is configured to receive the source streamlets **303** and generate the plurality of streams **202** of varying qualities. The original content file **200** from the publisher may be digital in form and may comprise content having a high bit rate such as, for example, 2 mbps. The content may be transferred from the publisher **110** to the content module **112** over the Internet **106**. Such transfers of data are well known in the art and do not require further discussion herein. Alternatively, the content may comprise a captured broadcast.

In a further embodiment, the encoding module **406** is configured to generate a plurality of sets **306** of streamlets **304**. The sets **306**, as described above with reference to FIG. **3***b*, may comprise streamlets having an identical time index and duration, and a unique bitrate. As with FIG. **3***b*, the sets **306** and subsequently the plurality of streams **202** may comprise the low quality stream **204**, the medium quality stream **206**, and the high quality stream **208**. Alternatively, the plurality of streams **202** may comprise any number of streams deemed necessary to accommodate end user bandwidth.

The encoder module **406** is further configured to encode each source streamlet **303** into the plurality of streams **202** and streamlet sets **306** and store the streamlets in the streamlet database **408**. The encoding module **406** may utilize encoding schemes such as DivX®, Windows Media Video 9®, Quicktime 6.5 Sorenson 3®, or Quicktime 6.5/MPEG-4®. Alternatively, a custom encoding scheme may be employed.

The content module **112** may also include a metadata module **412** and a metadata database **414**. In one embodiment, metadata comprises static searchable content information. For example, metadata includes, but is not limited to, air date of the content, title, actresses, actors, length, and episode name. Metadata is generated by the publisher **110**, and may be configured to define an end user environment. In one embodiment, the publisher **100** may define an end user navigational environment for the content including menus, thumbnails, sidebars, advertising, etc. Additionally, the publisher **110** may define functions such as fast forward, rewind, pause, and play that may be used with the content file **200**. The metadata module **412** is configured to receive the metadata from the publisher **110** and store the metadata in the metadata database **414**. In a further embodiment, the metadata module **412** is configured to interface with the client module **114**, allowing the client module **114** to search for content based upon at least one of a plurality of metadata criteria. Additionally, metadata may be generated by the content module **112** through automated process(es) or manual definition.

Once the streamlets **304** have been received and processed, the client module **114** may request streamlets **304** using HTTP from the web server **116**. Using a standard protocol such as HTTP eliminates the need for network administrators to configure firewalls to recognize and pass through network traffic for a new, specialized protocol. Additionally, since the client module **114** initiates the request, the web server **116** is only required to retrieve and serve the requested streamlet **304**. In a further embodiment, the client module **114** may be configured to retrieve streamlets **304** from a plurality of web servers **116**.

Each web server **116** may be located in various locations across the Internet **106**. The streamlets **304** may essentially be static files. As such, no specialized media server or server-side intelligence is required for a client module **114** to retrieve streamlets **304**. Streamlets **304** may be served by the web server **116** or cached by cache servers of Internet Service Providers (ISPs), or any other network infrastructure operators, and served by the cache server. Use of cache servers is well known to those skilled in the art, and will not be discussed further herein. Thus, a highly scalable solution is provided that is not hindered by massive amounts of client module **114** requests to the web server **116** at any specific location, especially the web server **116** most closely associated with or within the content module **112**

FIG. **5***a* is a schematic block diagram illustrating one embodiment of an encoder module **406** in accordance with the present invention. In one embodiment, the encoder module **406** may include a master module **502** and a plurality of host computing modules (hereinafter "host") **504**. The hosts **504** may comprise personal computers, servers, etc. In a further embodiment, the hosts **504** may be dedicated hardware, for example, cards plugged into a single computer.

The master module (hereinafter "master") **502** is configured to receive streamlets **303** from the streamlet module **404** and stage the streamlet **303** for processing. In one embodiment, the master **502** may decompress each source streamlet **303** to produce a raw streamlet. As used herein, the term "raw streamlet" refers to a streamlet **303** that is uncompressed or lightly compressed to substantially reduce size with no significant loss in quality. A lightly compressed raw streamlet can be transmitted more quickly and to more hosts. Each host **504** is coupled with the master **502** and configured to receive a raw streamlet from the master **502** for encoding. The hosts **504**, in one example, generate a plurality of streamlets **304** having identical time indices and durations, and varying bitrates. Essentially each host **504** may be configured to generate a set **306** from the raw streamlet **503** sent from the master **502**. Alternatively, each host **504** may be dedicated to producing a single bitrate in order to reduce the time required for encoding.

Upon encoding completion, the host **504** returns the set **306** to the master **502** so that the encoding module **406** may store the set **306** in the streamlet database **408**. The master **502** is further configured to assign encoding jobs to the hosts **504**. Each host is configured to submit an encoding job completion bid (hereinafter "bid"). The master **502** assigns encoding jobs depending on the bids from the hosts **504**. Each host **504** generates a bid depending upon a plurality of computing variables which may include, but are not limited to, current encoding job completion percentage, average job completion time, processor speed and physical memory capacity.

For example, a host **504** may submit a bid that indicates that based on past performance history the host **504** would be able to complete the encoding job in 15 seconds. The master **502** is configured to select from among a plurality of bids the best bid and subsequently submit the encoding job to the host **504** with the best bid. As such, the described encoding system does not require that each host **504** have identical hardware but beneficially takes advantage of the available computing power of the hosts **504**. Alternatively, the master **502** selects the host **504** based on a first come first serve basis, or some other algorithm deemed suitable for a particular encoding job.

The time required to encode one streamlet **304** is dependent upon the computing power of the host **504**, and the encoding requirements of the content file **200**. Examples of encoding requirements may include, but are not limited to, two or multi-pass encoding, and multiple streams of different bitrates. One benefit of the present invention is the ability to perform two-pass encoding on a live content file **200**. Typically, in order to perform two-pass encoding prior art systems must wait for the content file to be completed before encoding.

The present invention, however, segments the content file **200** into source streamlets **303** and the two-pass encoding to a plurality of streams **202** may be performed on each corresponding raw streamlet without waiting for a TV show to end, for example. As such, the content module **112** is capable of streaming the streamlets over the Internet shortly after the content module **112** begins capture of the content file **200**. The delay between a live broadcast transmitted from the publisher **110** and the availability of the content depends on the computing power of the hosts **504**.

FIG. **5***b* is a schematic block diagram illustrating one embodiment of parallel encoding of streamlets in accordance with the present invention. In one example, the capture module **402** (of FIG. **4**) begins to capture the content file and the streamlet module **404** generates a first streamlet **303***a* and passes the streamlet to the encoding module **406**. The encoding module **406** may take 10 seconds, for example, to generate the first set **306***a* of streamlets **304***a* (**304***a***1**, **304***a***2**, **304***a***3**, etc. represent streamlets **304** of

different bitrates). FIG. **5***b* illustrates the encoding process generically as block **502** to graphically illustrate the time duration required to process a raw or lightly encoded streamlet **303** as described above with reference to the encoding module **406**. The encoding module **406** may simultaneously process more than one streamlet **303**, and processing of streamlets will begin upon arrival of the streamlet from the capture module **402**.

During the 10 seconds required to encode the first streamlet **303***a*, the streamlet module **404** has generated five additional 2-second streamlets **303***b*, **303***c*, **303***d*, **303***e*, **303***f*, for encoding and the master **502** has prepared and staged the corresponding raw streamlets. Two seconds after the first set **306***a* is available the next set **306***b* is available, and so on. As such, the content file **200** is encoded for streaming over the Internet and appears live. The 10 second delay is given herein by way of example only. Multiple hosts **504** may be added to the encoding module **406** in order to increase the processing capacity of the encoding module **406**. The delay may be shortened to an almost unperceivable level by the addition of high CPU powered systems, or alternatively multiple low powered systems.

A system as described above beneficially enables multi-pass encoding of live events. Multi-pass encoding systems of the prior art require that the entire content be captured (or be complete) because in order to perform multi-pass encoding the entire content must be scanned and processed more than once. This is impossible with prior art systems because content from a live event is not complete until the event is over. As such, with prior art systems, multi-pass encoding can only be performed once the event is over. Streamlets, however, may be encoded as many times as is deemed necessary. Because the streamlet is an encapsulated media object of 2 seconds (for example), multi-pass encoding may begin on a live event once the first streamlet is captured. Shortly after multi-pass encoding of the first streamlet **303***a* is finished, multi-pass encoding of the second streamlet **303***b* finishes, and as such multi-pass encoding is performed on a live event and appears live to a viewer.

Any specific encoding scheme applied to a streamlet may take longer to complete than the time duration of the streamlet itself, for example, a very high quality encoding of a 2-second streamlet may take 5 seconds to finish. Alternatively, the processing time required for each streamlet may be less than the time duration of a streamlet. However, because the offset parallel encoding of successive streamlets are encoded by the encoding module at regular intervals (matching the intervals at which the those streamlets are submitted to the encoding module **406**, for example 2 seconds) the output timing of the encoding module **406** does not fall behind the real-time submission rate of the unencoded streamlets. Conversely, prior art encoding systems rely on the very fastest computing hardware and software because the systems must generate the output immediately in lock-step with the input. A prior art system that takes 2.1 seconds to encode 2 seconds worth of content is considered a failure. The present invention allows for slower than real-time encoding processes yet still achieves a real-time encoding effect due to the parallel offset pipes.

The parallel offset pipeline approach described with reference to FIG. **5***b* beneficially allows for long or short encoding times without "falling behind" the live event. Additionally, arbitrarily complex encoding of streamlets to multiple profiles and optimizations only lengthens the encoding time **502** without a perceptible difference to a user because the sets **306** of streamlets **304** are encoded in a time-selective manner so that streamlets are processed at regular time intervals and transmitted at these time intervals.

Returning now to FIG. **5***a*, as depicted, the master **502** and the hosts **504** may be located within a single local area network, or in other terms, the hosts **504** may be in close physical proximity to the master **502**. Alternatively, the hosts **504** may receive encoding jobs from the master **502** over the Internet or other communications network. For example, consider a live sports event in a remote location where it would be difficult to setup multiple hosts. In this example, a master performs no encoding or alternatively light encoding before publishing the streamlets online. The hosts **504** would then retrieve those streamlets and encode the streamlets into the multiple bitrate sets **306** as described above.

Furthermore, hosts **504** may be dynamically added or removed from the encoding module without restarting the encoding job and/or interrupting the publishing of streamlets. If a host **504** experiences a crash or some failure, its encoding work is simply reassigned to another host.

The encoding module **406**, in one embodiment, may also be configured to produce streamlets that are specific to a particular playback platform. For example, for a single raw streamlet, a single host **504** may produce streamlets for different quality levels for personal computer playback, streamlets for playback on cell phones with a different, proprietary codec, a small video-only streamlet for use when playing just a thumbnail view of the stream (like in a programming guide), and a very high quality streamlet for use in archiving.

FIG. **6***a* is a schematic block diagram illustrating one embodiment of a virtual timeline **600** in accordance with the present invention. In one embodiment, the virtual timeline **600** comprises at least one quantum media extension **602**. The quantum media extension (hereinafter "QMX") **602** describes an entire content file **200**. Therefore, the virtual timeline (hereinafter "VT") **600** may comprise a file that is configured to define a playlist for a user to view. For example, the VT may indicate that the publisher desires a user to watch a first show QMX **602***a* followed by QMX **602***b* and QMX **602***c*. As such, the publisher may define a broadcast schedule in a manner similar to a television station.

FIG. **6***b* is a schematic block diagram illustrating an alternative embodiment of a VT **600** in accordance with the present invention. In the depicted embodiment, the VT **600** may include a single QMX **602** which indicates that the publisher desires the same content to be looped over and over again. For example, the publisher may wish to broadcast a never-ending infomercial on a website.

FIG. **6***c* is a schematic block diagram illustrating one embodiment of a QMX **602** in accordance with the present invention. In one embodiment, the QMX **602** contains a multitude of information generated by the content module **112** configured to describe the content file **200**. Examples of information include, but are not limited to, start index **604**, end index **606**, whether the content is live **608**, proprietary publisher data **610**, encryption level **612**, content duration **614** and bitrate values **616**. The bitrate values **616** may include frame size **618**, audio channel **620** information, codecs **622** used, sample rate **624**, and frames parser **626**.

A publisher may utilize the QVT **600** together with the QMX **602** in order to prescribe a playback order for users, or alternatively selectively edit content. For example, a publisher may indicate in the QMX **602** that audio should be muted at time index 10:42 or video should be skipped for 3 seconds at time index 18:35. As such, the publisher may

selectively skip offensive content without the processing requirements of editing the content.

FIG. **7** is a schematic block diagram graphically illustrating one embodiment of a client module **114** in accordance with the present invention. The client module **114** may comprise an agent controller module **702**, a streamlet cache module **704**, and a network controller module **706**. In one embodiment, the agent controller module **702** is configured to interface with a viewer **708**, and transmit streamlets **304** to the viewer **708**. Alternatively, the agent controller module **702** may be configured to simply reassemble streamlets into a single file for transfer to an external device such as a portable video player.

In a further embodiment, the client module **114** may comprise a plurality of agent controller modules **702**. Each agent controller module **702** may be configured to interface with one viewer **708**. Alternatively, the agent controller module **702** may be configured to interface with a plurality of viewers **708**. The viewer **708** may be a media player (not shown) operating on a PC or handheld electronic device.

The agent controller module **702** is configured to select a quality level of streamlets to transmit to the viewer **708**. The agent controller module **702** requests lower or higher quality streams based upon continuous observation of time intervals between successive receive times of each requested streamlet. The method of requesting higher or lower quality streams will be discussed in greater detail below with reference to FIG. **10**.

The agent controller module **702** may be configured to receive user commands from the viewer **708**. Such commands may include play, fast forward, rewind, pause, and stop. In one embodiment, the agent controller module **702** requests streamlets **304** from the streamlet cache module **704** and arranges the received streamlets **304** in a staging module **709**. The staging module **709** may be configured to arrange the streamlets **304** in order of ascending playback time. In the depicted embodiment, the streamlets **304** are numbered 0, 1, 2, 3, 4, etc. However, each streamlet **304** may be identified with a unique filename.

Additionally, the agent controller module **702** may be configured to anticipate streamlet **304** requests and pre-request streamlets **304**. By pre-requesting streamlets **304**, the user may fast-forward, skip randomly, or rewind through the content and experience no buffering delay. In a further embodiment, the agent controller module **702** may request the streamlets **304** that correspond to time index intervals of 30 seconds within the total play time of the content. Alternatively, the agent controller module **702** may request streamlets at any interval less than the length of the time index. This enables a "fast-start" capability with no buffering wait when starting or fast-forwarding through content file **200**. In a further embodiment, the agent controller module **702** may be configured to pre-request streamlets **304** corresponding to specified index points within the content or within other content in anticipation of the end user **104** selecting new content to view. In one embodiment, the streamlet cache module **704** is configured to receive streamlet **304** requests from the agent controller module **702**. Upon receiving a request, the streamlet cache module **704** first checks a streamlet cache **710** to verify if the streamlet **304** is present. In a further embodiment, the streamlet cache module **704** handles streamlet **304** requests from a plurality of agent controller modules **702**. Alternatively, a streamlet cache module **704** may be provided for each agent controller module **702**. If the requested streamlet **304** is not present in the streamlet cache **410**, the request is passed to the network controller module **706**. In order to enable fast forward and rewind capabilities, the streamlet cache module **704** is configured to store the plurality of streamlets **304** in the streamlet cache **710** for a specified time period after the streamlet **304** has been viewed. However, once the streamlets **304** have been deleted, they may be requested again from the web server **116**.

The network controller module **706** may be configured to receive streamlet requests from the streamlet cache module **704** and open a connection to the web server **116** or other remote streamlet **304** database (not shown). In one embodiment, the network controller module **706** opens a TCP/IP connection to the web server **116** and generates a standard HTTP GET request for the requested streamlet **304**. Upon receiving the requested streamlet **304**, the network controller module **706** passes the streamlet **304** to the streamlet cache module **704** where it is stored in the streamlet cache **710**. In a further embodiment, the network controller module **706** is configured to process and request a plurality of streamlets **304** simultaneously. The network controller module **706** may also be configured to request a plurality of streamlets, where each streamlet **304** is subsequently requested in multiple parts.

In a further embodiment, streamlet requests may comprise requesting pieces of any streamlet file. Splitting the streamlet **304** into smaller pieces or portions beneficially allows for an increased efficiency potential, and also eliminates problems associated with multiple full-streamlet requests sharing the bandwidth at any given moment. This is achieved by using parallel TCP/IP connections for pieces of the streamlets **304**. Consequently, efficiency and network loss problems are overcome, and the streamlets arrive with more useful and predictable timing.

In one embodiment, the client module **114** is configured to use multiple TCP connections between the client module **114** and the web server **116** or web cache. The intervention of a cache may be transparent to the client or configured by the client as a forward cache. By requesting more than one streamlet **304** at a time in a manner referred to as "parallel retrieval," or more than one part of a streamlet **304** at a time, efficiency is raised significantly and latency is virtually eliminated. In a further embodiment, the client module allows a maximum of three outstanding streamlet **304** requests. The client module **114** may maintain additional open TCP connections as spares to be available should another connection fail. Streamlet **304** requests are rotated among all open connections to keep the TCP flow logic for any particular connection from falling into a slow-start or close mode. If the network controller module **706** has requested a streamlet **304** in multiple parts, with each part requested on mutually independent TCP/IP connections, the network controller module **706** reassembles the parts to present a complete streamlet **304** for use by all other components of the client module **114**.

When a TCP connection fails completely, a new request may be sent on a different connection for the same streamlet **304**. In a further embodiment, if a request is not being satisfied in a timely manner, a redundant request may be sent on a different connection for the same streamlet **304**. If the first streamlet request's response arrives before the redundant request response, the redundant request can be aborted. If the redundant request response arrives before the first request response, the first request may be aborted.

Several streamlet **304** requests may be sent on a single TCP connection, and the responses are caused to flow back in matching order along the same connection. This eliminates all but the first request latency. Because multiple responses are always being transmitted, the processing

latency of each new streamlet **304** response after the first is not a factor in performance. This technique is known in the industry as "pipelining." Pipelining offers efficiency in request-response processing by eliminating most of the effects of request latency. However, pipelining has serious vulnerabilities. Transmission delays affect all of the responses. If the single TCP connection fails, all of the outstanding requests and responses are lost. Pipelining causes a serial dependency between the requests.

Multiple TCP connections may be opened between the client module **114** and the web server **116** to achieve the latency-reduction efficiency benefits of pipelining while maintaining the independence of each streamlet **304** request. Several streamlet **304** requests may be sent concurrently, with each request being sent on a mutually distinct TCP connection. This technique is labeled "virtual pipelining" and is an innovation of the present invention. Multiple responses may be in transit concurrently, assuring that communication bandwidth between the client module **114** and the web server **116** is always being utilized. Virtual pipelining eliminates the vulnerabilities of traditional pipelining. A delay in or complete failure of one response does not affect the transmission of other responses because each response occupies an independent TCP connection. Any transmission bandwidth not in use by one of multiple responses (whether due to delays or TCP connection failure) may be utilized by other outstanding responses.

A single streamlet **304** request may be issued for an entire streamlet **304**, or multiple requests may be issued, each for a different part or portion of the streamlet. If the streamlet is requested in several parts, the parts may be recombined by the client module **114** streamlet.

In order to maintain a proper balance between maximized bandwidth utilization and response time, the issuance of new streamlet requests must be timed such that the web server **116** does not transmit the response before the client module **114** has fully received a response to one of the previously outstanding streamlet requests. For example, if three streamlet **304** requests are outstanding, the client module **114** should issue the next request slightly before one of the three responses is fully received and "out of the pipe." In other words, request timing is adjusted to keep three responses in transit. Sharing of bandwidth among four responses diminishes the net response time of the other three responses. The timing adjustment may be calculated dynamically by observation, and the request timing adjusted accordingly to maintain the proper balance of efficiency and response times.

The schematic flow chart diagrams that follow are generally set forth as logical flow chart diagrams. As such, the depicted order and labeled steps are indicative of one embodiment of the presented method. Other steps and methods may be conceived that are equivalent in function, logic, or effect to one or more steps, or portions thereof, of the illustrated method. Additionally, the format and symbols employed are provided to explain the logical steps of the method and are understood not to limit the scope of the method. Although various arrow types and line types may be employed in the flow chart diagrams, they are understood not to limit the scope of the corresponding method. Indeed, some arrows or other connectors may be used to indicate only the logical flow of the method. For instance, an arrow may indicate a waiting or monitoring period of unspecified duration between enumerated steps of the depicted method. Additionally, the order in which a particular method occurs may or may not strictly adhere to the order of the corresponding steps shown.

FIG. **8** is a schematic flow chart diagram illustrating one embodiment of a method **800** for processing content in accordance with the present invention. In one embodiment the method **800** starts **802**, and the content module **112** receives **804** content from the publisher **110**. Receiving content **804** may comprise receiving **804** a digital copy of the content file **200**, or digitizing a physical copy of the content file **200**. Alternatively, receiving **804** content may comprise capturing a radio, television, cable, or satellite broadcast. Once received **804**, the streamlet module **404** generates **808** a plurality of source streamlets **303** each having a fixed duration. Alternatively, the streamlets **303** may be generated with a fixed file size.

In one embodiment, generating **808** streamlets comprises dividing the content file **200** into a plurality of two second streamlets **303**. Alternatively, the streamlets may have any length less than or equal to the length of the stream **202**. The encoder module **406** then encodes **810** the streamlets **303** into sets **306** of streamlets **304**, in a plurality of streams **202** according to an encoding scheme. The quality may be predefined, or automatically set according to end user bandwidth, or in response to pre-designated publisher guidelines

In a further embodiment, the encoding scheme comprises a proprietary codec such as WMV9®. The encoder module **406** then stores **812** the encoded streamlets **304** in the streamlet database **408**. Once stored **812**, the web server **116** may then serve **814** the streamlets **304**. In one embodiment, serving **814** the streamlets **304** comprises receiving streamlet requests from the client module **114**, retrieving the requested streamlet **304** from the streamlet database **408**, and subsequently transmitting the streamlet **304** to the client module **114**. The method **800** then ends **816**.

FIG. **9** is a schematic flow chart diagram illustrating one embodiment of a method **900** for viewing a plurality of streamlets in accordance with the present invention. The method **900** starts and an agent controller module **702** is provided **904** and associated with a viewer **708** and provided with a staging module **709**. The agent controller module **702** then requests **906** a streamlet **304** from the streamlet cache module **704**. Alternatively, the agent controller module **702** may simultaneously request **906** a plurality of streamlets **304** the streamlet cache module **704**. If the streamlet is stored **908** locally in the streamlet cache **710**, the streamlet cache module **704** retrieves **910** the streamlet **304** and sends the streamlet to the agent controller module **702**. Upon retrieving **910** or receiving a streamlet, the agent controller module **702** makes **911** a determination of whether or not to shift to a higher or lower quality stream **202**. This determination will be described below in greater detail with reference to FIG. **10**.

In one embodiment, the staging module **709** then arranges **912** the streamlets **304** into the proper order, and the agent controller module **702** delivers **914** the streamlets to the viewer **708**. In a further embodiment, delivering **914** streamlets **304** to the end user comprises playing video and or audio streamlets on the viewer **708**. If the streamlets **304** are not stored **908** locally, the streamlet request is passed to the network controller module **706**. The network controller module **706** then requests **916** the streamlet **304** from the web server **116**. Once the streamlet **304** is received, the network controller module **706** passes the streamlet to the streamlet cache module **704**. The streamlet cache module **704** archives **918** the streamlet. Alternatively, the streamlet cache module **704** then archives **918** the streamlet and passes the streamlet to the agent controller module **702**, and the method **900** then continues from operation **910** as described above.

Referring now to FIG. **10**, shown therein is a schematic flow chart diagram illustrating one embodiment of a method **1000** for requesting streamlets **304** within an adaptive-rate shifting content streaming environment in accordance with the present invention. The method **1000** may be used in one embodiment as the operation **911** of FIG. **9**. The method **1000** starts and the agent controller module **702** receives **1004** a streamlet **304** as described above with reference to FIG. **9**. The agent controller module **702** then monitors **1006** the receive time of the requested streamlet. In one embodiment, the agent controller module **702** monitors the time intervals A between successive receive times for each streamlet response. Ordering of the responses in relation to the order of their corresponding requests is not relevant.

Because network behavioral characteristics fluctuate, sometimes quite suddenly, any given Δ may vary substantially from another. In order to compensate for this fluctuation, the agent controller module **702** calculates **1008** a performance ratio r across a window of n samples for streamlets of playback length S. In one embodiment, the performance ratio r is calculated using the equation:

$$r = S\frac{n}{\sum_{i=1}^{n} \Delta_i}$$

Due to multiple simultaneous streamlet processing, and in order to better judge the central tendency of the performance ratio r, the agent controller module **702** may calculate a geometric mean, or alternatively an equivalent averaging algorithm, across a window of size in, and obtain a performance factor φ:

$$\varphi_{current} = \left(\prod_{j=1}^{m} r_j\right)^{\frac{1}{m}}$$

The policy determination about whether or not to upshift **1010** playback quality begins by comparing $\varphi_{current}$ with a trigger threshold Θ_{up}. If $\varphi_{current} \geq \Theta_{up}$, then an up shift to the next higher quality stream may be considered **1016**. In one embodiment, the trigger threshold Θ_{up} is determined by a combination of factors relating to the current read ahead margin (i.e. the amount of contiguously available streamlets that have been sequentially arranged by the staging module **709** for presentation at the current playback time index), and a minimum safety margin. In one embodiment, the minimum safety margin may be 24 seconds. The smaller the read ahead margin, the larger Θ_{up} is to discourage upshifting until a larger read ahead margin may be established to withstand network disruptions. If the agent controller module **702** is able to sustain **1016** upshift quality, then the agent controller module **702** will upshift **1017** the quality and subsequently request higher quality streams. The determination of whether use of the higher quality stream is sustainable **1016** is made by comparing an estimate of the higher quality stream's performance factor, φ_{higher}, with Θ_{up}. If $\varphi_{higher} \geq \Theta_{up}$ then use of the higher quality stream is considered sustainable. If the decision of whether or not the higher stream rate is sustainable **1016** is "no," the agent controller module **702** will not attempt to upshift **1017** stream quality. If the end of the stream has been reached **1014**, the method **1000** ends **1016**.

If the decision on whether or not to attempt upshift **1010** is "no", a decision about whether or not to downshift **1012** is made. In one embodiment, a trigger threshold Θ_{down} is defined in a manner analogous to Θ_{up}. If $\varphi_{current} > \Theta_{down}$ then the stream quality may be adequate, and the agent controller module **702** does not downshift **1018** stream quality. However, if $\varphi_{current} \leq \Theta_{down}$, the agent controller module **702** does downshift **1018** the stream quality. If the end of the stream has not been reached **1014**, the agent controller module **702** begins to request and receive **1004** lower quality streamlets and the method **1000** starts again. Of course, the above described equations and algorithms are illustrative only, and may be replaced by alternative streamlet monitoring solutions.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. A system for adaptive-rate content streaming of live event video playable on one or more end user stations over the Internet, the system comprising:

at least one storage device storing live event video, the live event video encoded at a plurality of different bitrates creating a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, the low quality stream, the medium quality stream, and the high quality stream each comprising a group of streamlets encoded at a respective one of the plurality of different bitrates;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps;

wherein the amount of data in each streamlet corresponding to the same part of the live event video in the low quality stream, the medium quality stream, and the high quality stream varies according to the different bitrates; and

the plurality of streamlets in the low quality stream, the medium quality stream, and the high quality stream having a duration that is the same as each other.

2. The system of claim **1**, wherein the low quality stream is encoded at a bit rate of no greater than 100 kbps, and the medium quality stream is encoded at a bit rate between 100 kbps and 600 kbps.

3. The system of claim **1** wherein the streamlets in each of the high quality stream, the medium quality stream and the low quality stream are each encoded at a different one of the plurality of different bitrates.

4. The system of claim **1**, further comprising:

a plurality of web servers located at different locations across the internet, each web server configured to:

receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing a portion of the video, wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams;

retrieve from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

send the retrieved first streamlet from the currently selected one of the different copies to the requesting one of the end user stations over the one or more network connections.

5. The system of claim **1**, wherein each of the first streamlets has a first duration that is the range of 0.1 to 5 seconds.

6. The system of claim **1**, wherein the live event is a live sports event.

7. The system of claim **1**, further comprising:

a first web server configured to:

receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the live event video, wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the live event video;

retrieve from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

send the retrieved first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream to the requesting one of the end user stations over the one or more network connections.

8. The system of claim **7**, wherein the first streamlets of the low quality stream, the medium quality stream, and the high quality stream are available before the live event is complete.

9. The system of claim **7**, wherein the streamlets of the low quality stream, the medium quality stream, and the high quality stream of the live event are available on a 10 second delay.

10. A content player device to stream a video over a network from a server for playback of the video, the content player device comprising:

a processor;

a digital processing apparatus memory device comprising non-transitory machine-readable instructions that, when executed, cause the processor to:

establish one or more network connections between the client module and the server, wherein the server is configured to access at least one of a plurality of groups of streamlets;

wherein the video is encoded at a plurality of different bitrates to create a plurality of streams including at least a low quality stream, a medium quality stream, and a high quality stream, wherein each of the low quality stream, the medium quality stream, and the high quality stream comprises a streamlet that encodes the same portion of the video at a different one of the plurality of different bitrates;

wherein at least one of the low quality stream, medium quality stream, and high quality stream is encoded at a bit rate of no less than 600 kbps; and

wherein the streamlet encoding the same portion of the video in the low quality stream has an equal playback duration as the streamlet encoding the same portion of the video in the high quality stream;

select a specific one of the streams based upon a determination by the client module to select a higher or lower bitrate version of the streams;

place a streamlet request to the server over the one or more network connections for the selected stream;

receive the requested streamlets from the server via the one or more network connections; and

provide the received streamlets for playback of the video.

11. The content player device of claim **10** wherein each streamlet of the plurality of streamlets in the low quality stream, the medium quality stream, and the high quality stream has a duration that is the same as each other.

12. The content player device of claim **10**, wherein the low quality stream is encoded at a bit rate of no greater than 100 kbps and the medium quality stream is encoded at a bit rate between 100 kbps and 600 kbps.

13. The content player device of claim **10**, wherein each streamlet of the plurality of streamlets in the low quality stream, the medium quality stream, and the high quality stream has a duration that is the range of 0.1 to 5 seconds.

14. The content player device of claim **10**, wherein the video is a video of a live event.

15. The content player device of claim **14**, wherein the streamlets of the low quality stream, the medium quality stream, and the high quality stream are available before the live event is complete.

16. The content player device of claim **15**, wherein the streamlets of the low quality stream, the medium quality stream, and the high quality stream of the live event are available on a ten second delay.

17. The content player device of claim **16**, wherein the streamlets from the low quality stream, the medium quality stream, and the high quality stream of the live event, when played back, appear live to a viewer.

18. A system for adaptive-rate content streaming of live event video playable on one or more end user stations over the internet, the system comprising:

at least one storage device configured to store live event video, the live event video encoded at a plurality of different bit rates creating a plurality of streams including at least a low quality stream and a high quality stream;

the low quality stream and the high quality stream each encoding the same portion of the live event video with a streamlet that is encoded a different one of the different bit rates;

wherein the plurality of streamlets in the low quality stream and the plurality of streamlets in the high quality stream have durations that are equal to each other.

19. The system of claim **18**, wherein the streamlets in each of the low quality stream and the high quality stream corresponding to the same portion of the live event video have equal durations.

20. The system of claim **18** wherein the plurality of streams further comprise a medium quality stream encoded at a bit rate higher than the low quality stream and lower than the high quality stream.

21. The system of claim **20** wherein the low quality stream is encoded at a bit rate of no less than 100 kbps, the high quality stream is encoded at a bit rate of no less than 600

kbps, and the medium quality stream is encoded at a bit rate between 100 kbps and 600 kbps.

22. The system of claim **18**, further comprising:

a first web server configured to:

receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve a streamlet storing a portion of the video, wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the client module to select a higher or lower bitrate version of the streams;

retrieve from the storage device the requested streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

send the retrieved streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream to the requesting one of the end user stations over the one or more network connections.

23. The system of claim **18**, further comprising:

a plurality of web servers located at different locations across the internet, each web server configured to:

receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve a streamlet storing a portion of the video, wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the client module to select a higher or lower bitrate version of the streams;

retrieve from the storage device the requested streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

send the retrieved streamlet from the currently selected one of the different copies to the requesting one of the end user stations over the one or more network connections.

24. The system of claim **21**, wherein all of the streamlets in each of the low quality stream, the medium quality stream and the high quality stream have equal durations.

25. The system of claim **21**, wherein the streamlets of the low quality stream, the medium quality stream, and the high quality stream are available on a ten second delay before the live event is complete, wherein the streamlets from the low quality stream, the medium quality stream, and the high quality stream of the live event, when played back, appear live to a viewer.

26. A content player device to stream a video over a network from a server for playback of the video, the content player device comprising:

a processor;

a digital processing apparatus memory device comprising non-transitory machine-readable instructions that, when executed, cause the processor to:

establish one or more network connections between the client module and the server, wherein the server is configured to access at least one of a plurality of groups of streamlets;

wherein the video is encoded at a plurality of different bitrates to create a plurality of streams including at least a low quality stream and a high quality stream,

the low quality stream and the high quality stream each representing the same portion of the video with a streamlets encoded at a different one of the plurality of different bitrates; and

wherein the streamlet representing the same portion of the video in the low quality stream and the streamlet representing the same portion of the video in the high quality stream have durations equal to each other;

select a specific one of the streams based upon a determination by the client module to select a higher or lower bitrate version of the streams;

place a streamlet request to the server over the one or more network connections for the selected stream;

receive the requested streamlets from the server via the one or more network connections; and

provide the received streamlets for playback of the video.

27. The content player device of claim **26**, wherein the plurality of streamlets in the low quality stream have a duration equal to the duration of the plurality of streamlets in the high quality stream.

* * * * *

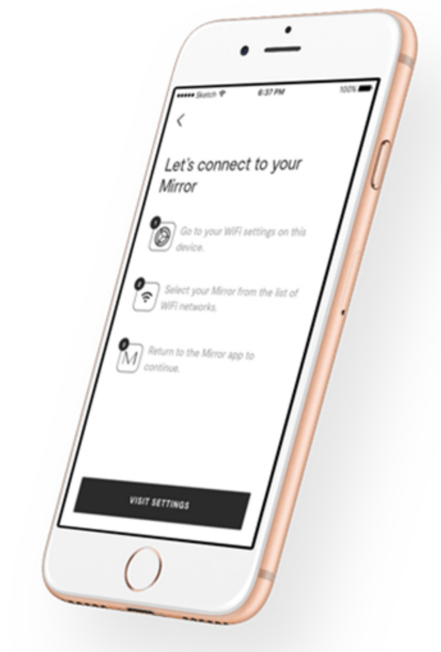
EXHIBIT C-1

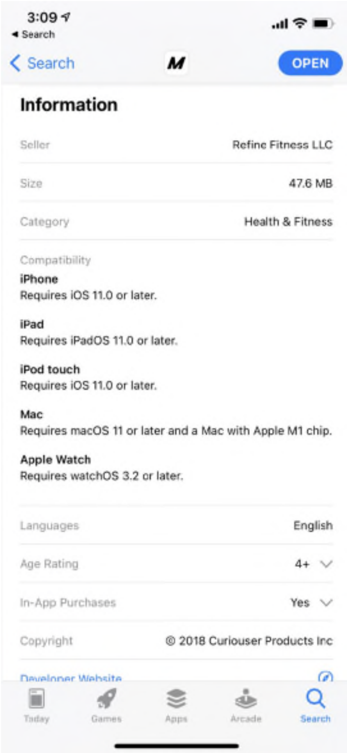
U.S. Patent No. 10,469,555 to Mirror

The following claim chart shows exemplary aspects of the Mirror Application and Mirror Device that infringe the claims below. The chart is exemplary and should not be read to limit DISH's claims against Mirror to the specific products or services described below. The chart should also not be read to limit DISH's claims to the patent claims charted below. Nor should the chart below be read to limit how the Mirror Application and Mirror Devices infringe the claims below.

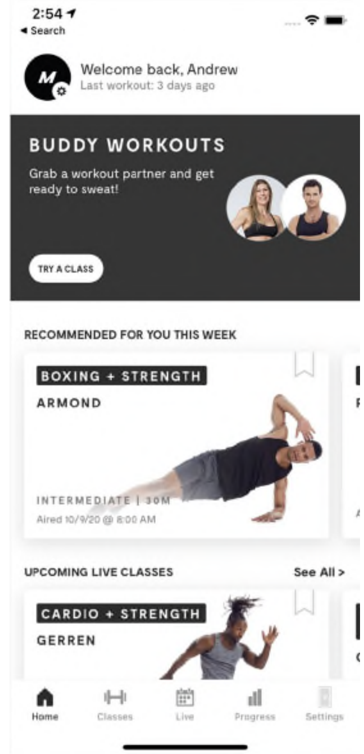
Claim	Claim Limitation	Example Infringement Evidence
10	A content player device to stream a video over a network from a server for playback of the video, the content player device comprising:	The Mirror Application is software that permits an end user station having "a content player device to stream a video over a network from a server for playback of the video." The Mirror Application is executable by end user stations that have a content player device and it obtains streams of a selected video program for playback. The streams are obtained by the Mirror Application over a network. The exemplary images in this chart of the Mirror Application are from the Mirror Application running on an Apple iPhone XS (Mirror's iOS Application). In addition, the Mirror Application is available to run on other devices Except as otherwise noted, each of these devices is an end user station with a "content player device." **MIRROR DIGITAL OVERVIEW** Access all of MIRROR's Live and On Demand Classes 24/7 from your phone, tablet, and smart TV devices. This feature is included for free with the MIRROR Membership and access is available immediately (whether you already have your Mirror or are awaiting delivery). Please note, it is not currently possible to purchase MIRROR Digital if you do not own a Mirror. https://mirror.kustomer.help/en_us/mirror-digital-overview-B1pnVd8UU

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		GET THE MIRROR APP To get started setting up your Mirror, you need to download the MIRROR app from the Apple App Store or Google Play Store. The app will take you through everything you need to know. Available on the App Store Get it on Google play Need help? Email us at hello@mirror.co  https://www.mirror.co/app. MIRROR APP The MIRROR App allows you to access and customize the Mirror experience. The MIRROR App is available for both iOS and Android! • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-app--S1dDC_tYm.

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Apple App Store When launched, the Mirror Application displays a main menu:

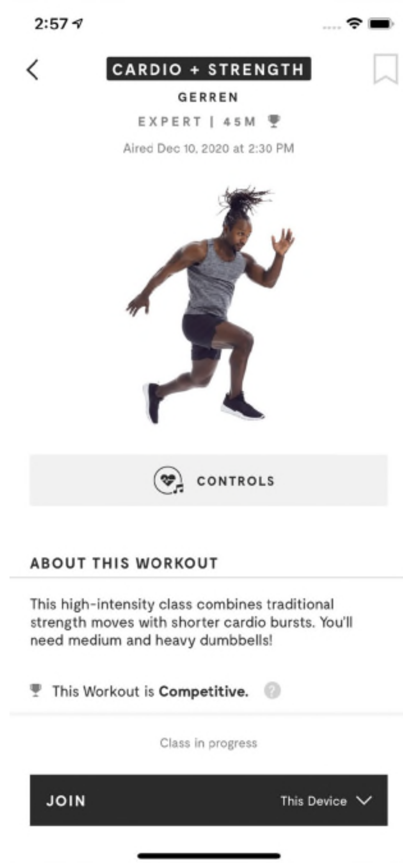
USP 10,469,555 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>


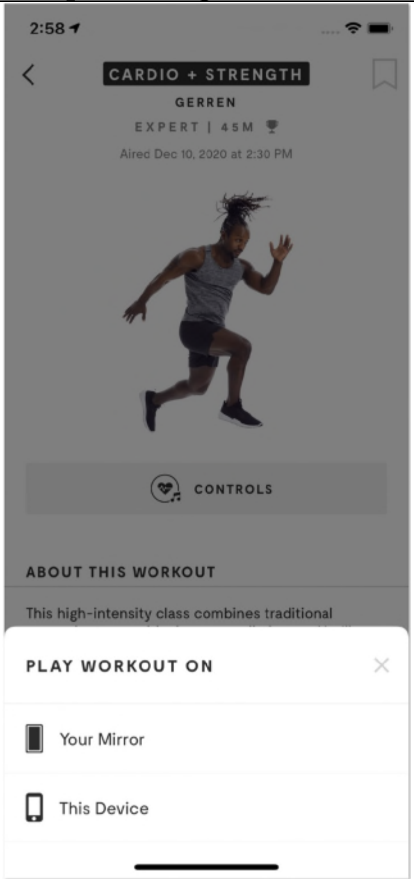
<u>Source: Mirror iOS Application</u>

The main menu of the Mirror Application displays on-demand and live classes that are each “a video.” The “Live” section of the Mirror Application main menu displays a preview of ongoing and upcoming live videos. The “Classes” section of the Mirror Application main menu displays a preview of on-demand videos. Selecting a video causes the video to stream over the Internet and playback on the Mirror Application. Selecting a video causes the Mirror Application to provide options to stream the class to a variety of end user stations having content players, including the iOS device that the Mirror Application is executing on or the separate Mirror Device.
</td></tr>
</table>

USP 10,469,555 to Mirror

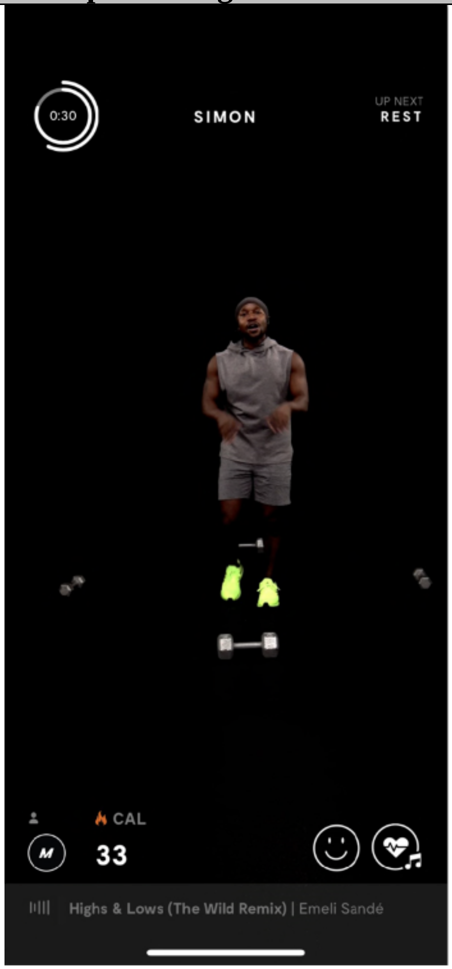
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting “Your Mirror” causes the video and other materials to be streamed on the user’s Mirror device, which is a content player device that is connected to the internet.

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Alternatively, selecting "This Device" causes the selected workout video and other materials to be streamed on the user's iOS device:

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application As shown above, Mirror Devices are end user stations having a "content player device to stream a video over a network from a server for playback of the video."

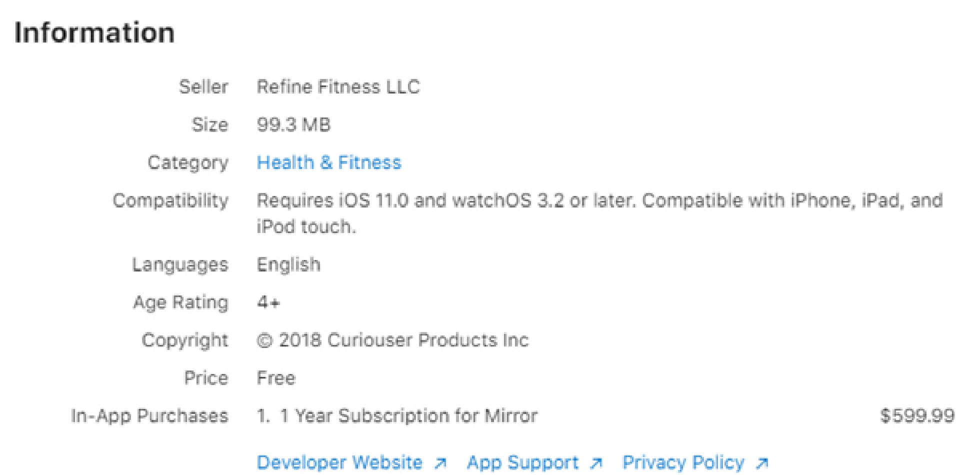
USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
	a processor;	A device running the Mirror Application includes at least one "processor" to execute the Mirror Application and stream the video. The devices that are compatible with the Mirror Application, each include one or more processors. **MIRROR DIGITAL COMPATIBLE DEVICES** The MIRROR App is available for the iPhone, iPad, Android phones, and Android tablets. MIRROR Digital can be casted to your smart TV using these devices. • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-digital-compatible-devices-HklDdOU8U. For example, Mirror requires users to provide a user device such as an iPhone that includes a processor to execute the Mirror Application **Information** Seller: Refine Fitness LLC Size: 99.3 MB Category: Health & Fitness Compatibility: Requires iOS 11.0 and watchOS 3.2 or later. Compatible with iPhone, iPad, and iPod touch. Languages: English Age Rating: 4+ Copyright: © 2018 Curiouser Products Inc Price: Free In-App Purchases: 1. 1 Year Subscription for Mirror $599.99 Developer Website ↗ App Support ↗ Privacy Policy ↗ https://apps.apple.com/us/app/mirror-workout-companion/id1153358600.

Claim	Claim Limitation	Example Infringement Evidence
		The Mirror Devices also include a processor. **HARDWARE** FRAME — Carbon steel frame; Mineral bronze powder coated DISPLAY — 40" full HD 1080p display, with 178° wide viewing angle TECHNOLOGY — Quad core processor SOUND — 2 x 10 watt high-fidelity stereo speakers; Embedded omnidirectional microphone CAMERA — 5 megapixel front-facing camera POWER — 1 ft and 6 ft right angle UL certified cables https://www.mirror.co/shop/mirror.
	a digital processing apparatus memory device comprising non-transitory	The device executes the Mirror Application from "a digital processing apparatus memory device comprising non-transitory machine-readable instructions." The instructions include at least the executable instructions for the Mirror Application and its features. Mirror requires users of the

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td>machine-readable instructions that, when executed, cause the processor to:</td><td>Mirror Application to provide a device with a digital processing apparatus memory device to store the instructions.

For example, Mirror requires users to provide at least 99.3 MB of storage on a digital processing apparatus memory device of the end user station for storing the Mirror Application.



https://apps.apple.com/us/app/mirror-workout-companion/id1153358600.

The Mirror Devices also include “a digital processing apparatus memory device comprising non-transitory machine-readable instructions.” For example, the on-board quad core processor requires memory containing non-transitory machine-readable instructions in order to process and display streaming fitness classes.</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		HARDWARE FRAME — Carbon steel frame Mineral bronze powder coated DISPLAY — 40" full HD 1080p display, with 178° wide viewing angle TECHNOLOGY — Quad core processor SOUND — 2 x 10 watt high-fidelity stereo speakers Embedded omnidirectional microphone CAMERA — 5 megapixel front-facing camera POWER — 1 ft and 6 ft right angle UL certified cables https://www.mirror.co/shop/mirror.
	establish one or more network connections between the client module and the server, wherein the server is configured	As shown below, the non-transitory machine-readable instructions of the Mirror Application and Mirror Devices, when executed, cause the processor(s) to “establish one or more network connections between the client module and the server” that is “configured to access at least one of a plurality of groups of streamlets.” The “segments” discussed herein are “streamlets.” The Mirror Application requires an internet connection.

Claim	Claim Limitation	Example Infringement Evidence
	to access at least one of a plurality of groups of streamlets;	**PRELOAD CLASSES ON MIRROR DIGITAL** You currently cannot preload classes on the MIRROR Digital, however this feature is coming soon! If you are not able to connect WiFi or are in a tough WiFi environment, you can always use cellular data to stream classes on the MIRROR App. Please consult your cell phone provider for questions about data usage and your plan. https://mirror.kustomer.help/en_us/preload-classes-on-mirror-app-H12XPdUUL. To stream a video, such as that shown above, the Mirror Application requests a stream of a selected video via a network connection. A user may select a video, as described above, the stream the video. When the Mirror Application accesses a selected video, it requests an receives a playlist file that shows the available versions of the video at different bandwidths resolutions. For the following test, a live video was selected. In the test, an iPhone 11 running the Mirror Application makes an HTTPS GET request for a master playlist named "**playlist.m3u8**" that specifies the available streams and provides links to the playlists for those streams. The following master playlist named "**playlist.m3u8**" is returned. #EXTM3U #EXT-X-VERSION:3 #EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080 ../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720 ../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480 ../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360 ../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288 ../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180 ../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 Filename: playlist.m3u8.

Claim	Claim Limitation	Example Infringement Evidence
		This is a master playlist file according to the HLS specification.[1] The playlist shows six versions of the stream, denoted by each #EXT-X-STREAM-INF tag at the following bandwidth: • 6434112 (referred to herein as "**6434112 Bandwidth**") • 864048 (referred to herein as "**864048 Bandwidth**") • 403824 (referred to herein as "**403824 Bandwidth**") • 367728 (referred to herein as "**367728 Bandwidth**") • 312832 (referred to herein as "**312832 Bandwidth**") • 249664 (referred to herein as "**249664 Bandwidth**") For each of these versions, the master playlist provides a link to a playlist file for the specified version of the selected video at a particular bandwidth and resolution, which is called a "variant" in HLS. The Mirror Application initially selects the **6434112 Bandwidth** (1080p – high bandwidth) version of the stream and makes a request for the corresponding variant playlist file named "**chunklist.m3u8**." That file is returned with the following contents (a portion of which is shown below).

[1] RFC 8216 (HLS Live Streaming), Section 4.3.4 (Master Playlist Tags)

Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1232 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:08.356Z #EXTINF:2.0, r4vhrugx/00000000/media_1232.ts #EXTINF:2.0, r4vhrugx/00000000/media_1233.ts #EXTINF:2.0, r4vhrugx/00000000/media_1234.ts #EXTINF:2.0, r4vhrugx/00000000/media_1235.ts #EXTINF:2.0, r4vhrugx/00000000/media_1236.ts #EXTINF:2.0, r4vhrugx/00000000/media_1237.ts #EXTINF:2.0, r4vhrugx/00000000/media_1238.ts #EXTINF:2.0, r4vhrugx/00000000/media_1239.ts #EXTINF:2.0, r4vhrugx/00000000/media_1240.ts #EXTINF:2.0, r4vhrugx/00000000/media_1241.ts #EXTINF:2.0, Filename: https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 As noted above, the variant playlist file is an HLS playlist. The variant playlist file identifies a plurality of segments or "streamlets" that are part of the **6434112 Bandwidth** group of segments. Each line in the file "**chunklist.m3u8**" that begins with "#EXTINF" specifies the length of the segments in seconds (2.0). The line below the #EXTINF entry is the relative location of the video file for the segment (e.g., r4vhrugx/0000000/media_1232.ts). The Mirror Application uses HTTPS GET requests to retrieve the

Claim	Claim Limitation	Example Infringement Evidence
		segments of the encoded video specified in the file above , which accesses the segments in response to requests from the Mirror Application. The Mirror Application makes the request for a segment media_1232.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected video. As long as the viewer stays on the selected video and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **6434112 Bandwidth** version). While the example above relates to the **6434112 Bandwidth** group of streamlets, other groups of streamlets are available. If the available bandwidth for the network connection decreases, for example, the Mirror Application will continue to request segments to continue streaming the video, but at one of the lower bandwidths. For example, for the current test, the Mirror Application subsequently made a request to for the corresponding variant playlist file for the **403824 Bandwidth** named "chunklist.m3u8." That file is returned with the following contents showing a portion of the **403824 Bandwidth** group of segments for the video being streamed.

Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1238 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z #EXTINF:2.0, fbd862nq/00000000/media_1238.ts #EXTINF:2.0, fbd862nq/00000000/media_1239.ts #EXTINF:2.0, fbd862nq/00000000/media_1240.ts #EXTINF:2.0, fbd862nq/00000000/media_1241.ts #EXTINF:2.0, fbd862nq/00000000/media_1242.ts #EXTINF:2.0, fbd862nq/00000000/media_1243.ts #EXTINF:2.0, fbd862nq/00000000/media_1244.ts #EXTINF:2.0, fbd862nq/00000000/media_1245.ts #EXTINF:2.0, fbd862nq/00000000/media_1246.ts #EXTINF:2.0, fbd862nq/00000000/media_1247.ts #EXTINF:2.0, fbd862nq/00000000/media_1248.ts Filename: **chunklist.m3u8** The Mirror Application then makes the request for media_1238.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected video at the **403824 Bandwidth**. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **403824 Bandwidth** version).

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>As the bandwidth is further constrained, the Mirror Application makes another request for a lower quality stream. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variable playlist file for the 249664 Bandwidth named “chunklist.m3u8.” That file is returned with the following contents showing a portion of the 249664 Bandwidth group of segments for the video being streamed.

#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1241
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:26.354Z
#EXTINF:2.0,
zf4q4ivl/00000000/media_1241.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1242.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1243.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1244.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1245.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1246.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1247.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1248.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1249.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1250.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1251.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1252.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1253.ts

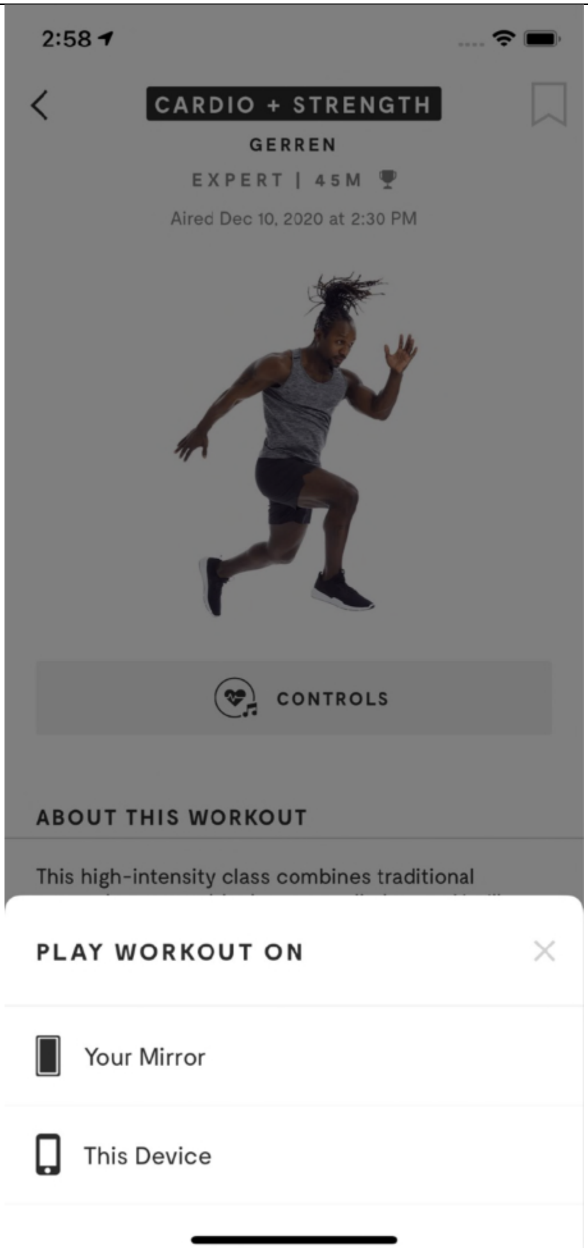
File: chunklist.m3u8</td></tr>
</table>

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		The Mirror Application then makes the request for media_1281.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application plays back the segment to stream the selected video at the **249664 Bandwidth**. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **249664 Bandwidth** version). A portion of a subsequently retrieved playlist's contents are shown below. #EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1245 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:34.354Z #EXTINF:2.0, zf4q4ivl/00000000/media_1245.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1246.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1247.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1248.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1249.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1250.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1251.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1252.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1253.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1254.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1255.ts #EXTINF:2.0,

Claim	Claim Limitation	Example Infringement Evidence
		Filename: **chunklist.m3u8** The subsequently retrieved playlist includes additional video segments that were not included in the previous playlist file, for example: "**media_1254.ts**" and "**media_1255.ts.**" The Mirror Application continues to request, receive, and playback successive segments of the video to stream the video. The Mirror Devices also require an internet connection . **CONNECTION** INTERNET — Dual-band 802.11 A/B/G/N Wi-Fi APP — Controlled by iOS or Android companion app HEART RATE — Syncs with Bluetooth™ heart rate monitors, Apple Watches, and Android Wear OS Watches AUDIO — Pairs with Bluetooth™ speakers and headphones https://www.mirror.co/shop/mirror To stream a video, such as that shown above, the Mirror Devices request a stream of a selected video program via a network connection. The iOS application provides the interface for interacting with a Mirror Device (i.e., selecting a live or on demand class to stream). After selecting a class, the user selects whether to use the iOS device or Mirror Device to view the content (i.e., stream and participate in the workout).

USP 10,469,555 to Mirror

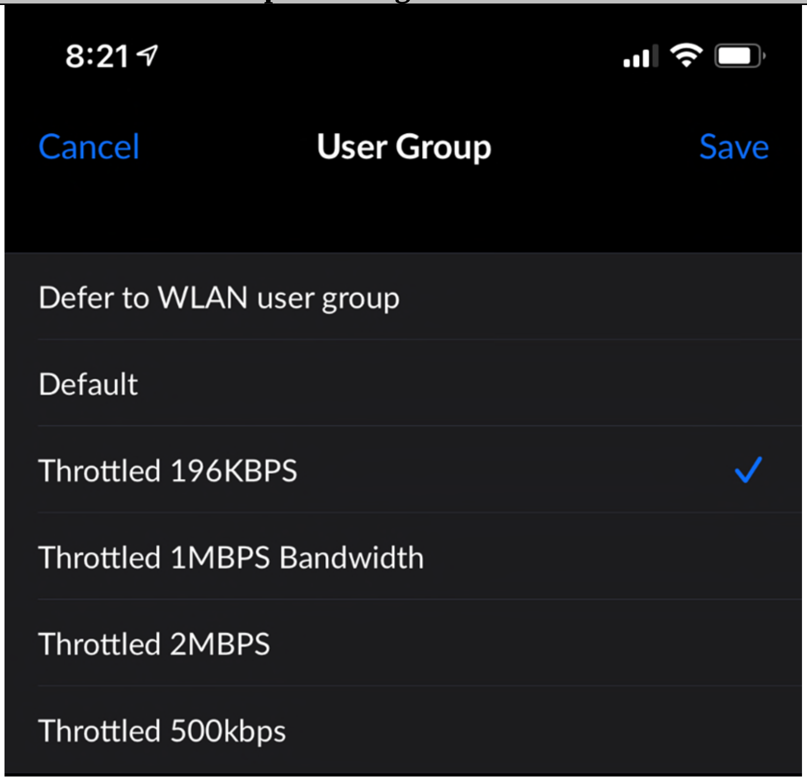
		

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Source: Mirror iOS Application Selecting "Your Mirror" causes the Mirror Device to initiate streaming requests:  For the following test, a live video was selected. Based on the test, and upon information and belief, the Mirror Devices make the same or substantially the same requests to and retrievals as shown above with respect to the Mirror Application. For example, when the Mirror Device(s) accesses a selected live event video, the Mirror Device(s) initially selects a first bandwidth version of the stream, makes a request for the segments of the group corresponding to the selected bandwidth version of the live event program, receives segments from

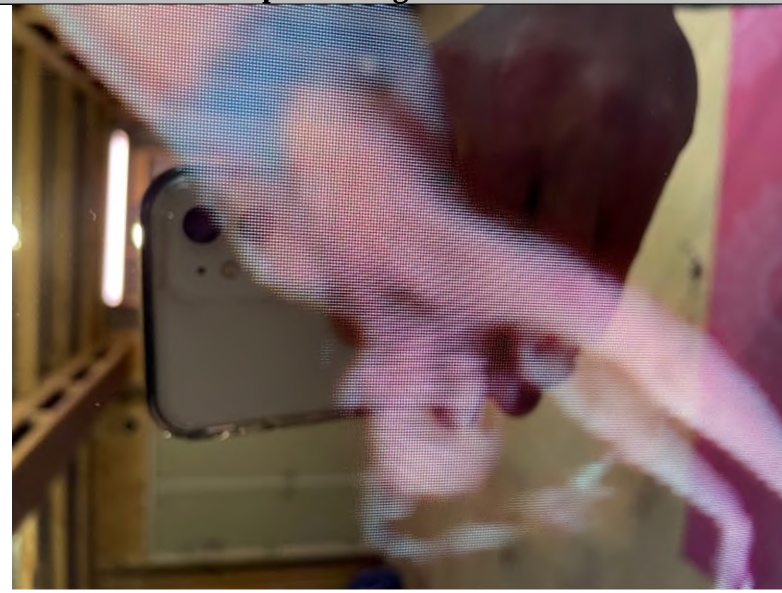
Claim	Claim Limitation	Example Infringement Evidence
		the group corresponding to the selected bandwidth version, and then plays the requested and received segments on the Mirror Device content player as shown below.  Other groups of streamlets are also available. For example, for the current test, bandwidth for the Mirror Device was constrained to 196Kbps, which caused the Mirror Device to display a "buffering" message while requesting and receiving a corresponding playlist and streamlets for a second bandwidth version of the live event video as shown below.

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 The image resolution for the second bandwidth streamlet requested is noticeably lower quality as indicated by the pixelated edges of the instructor’s body, as shown below.

Claim	Claim Limitation	Example Infringement Evidence
		
	wherein the video is encoded at a plurality of different bitrates to create a plurality of streams including at least a low quality stream, a medium-quality stream, and a high quality stream, wherein each of the low quality stream, the medium-quality stream, and the high quality stream comprises a streamlet	The "video is encoded at a plurality of different bitrates to create a plurality of streams." The plurality of different bitrates creates a plurality of streams "including at least a low quality stream, a medium quality stream, and a high quality stream." And "each of the low quality stream, the medium quality stream, and the high quality stream comprises a streamlet that encodes the same portion of the video at a different one of the plurality of bitrates. As shown in the master playlist file, "**playlist.m3u8**," the selected video is encoded at 6 different bitrates.

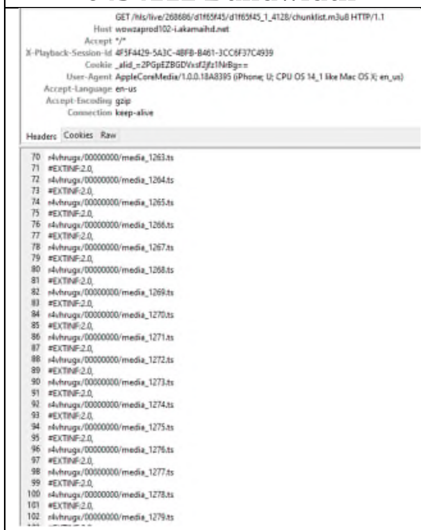
USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
	that encodes the same portion of the video at a different one of the plurality of different bitrates;	#EXTM3U #EXT-X-VERSION:3 #EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080 ../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720 ../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480 ../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360 ../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288 ../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180 ../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 Filename: playlist.m3u8. The different bitrates correspond to six different video streams of the same video content but at varying quality: • 6434112 (referred to herein as "**6434112 Bandwidth**") • 864048 (referred to herein as "**864048 Bandwidth**") • 403824 (referred to herein as "**403824 Bandwidth**") • 367728 (referred to herein as "**367728 Bandwidth**") • 312832 (referred to herein as "**312832 Bandwidth**") • 249664 (referred to herein as "**249664 Bandwidth**") These different bitrate versions include at least a "low quality stream, a medium quality stream, and a high quality stream." For example, the **6434112 Bandwidth** version can be considered a high-quality stream, the **403824 Bandwidth** version can be considered a medium-quality stream, and the **249664 Bandwidth** version can be considered a low-quality stream.

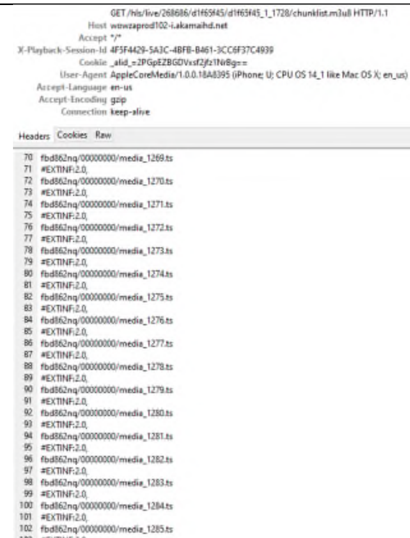
USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		As shown herein, each of the high-quality stream (e.g., the **6434112 Bandwidth** stream), the medium-quality stream (e.g., the **403824 Bandwidth** stream), and the low-quality stream (e.g., the **249664 Bandwidth** stream) comprise “a group of streamlets encoded at the same respective one of the different bitrates.” Each variant playlist includes multiple streamlets (e.g., ”comprises a streamlet”), including a “streamlet” with the filename ending in a **media_1275.ts**.” A comparison of the **6434112 Bandwidth**, **403823 Bandwidth**, and **249664 Bandwidth** versions from above shows that each playlist includes a segment filename ending with “media_1275.ts.” On information and belief, playlists for the other three variants also include this segment. As discussed above, each streamlet corresponds to a portion of the video. Notably, for example, each bitrate version of the **media_1275.ts** segment has a duration of 2.0 seconds (as indicated in each line beginning with “#EXTINF”), indicating that each “encodes the same portion of the video at a different one of the plurality of bitrates.”

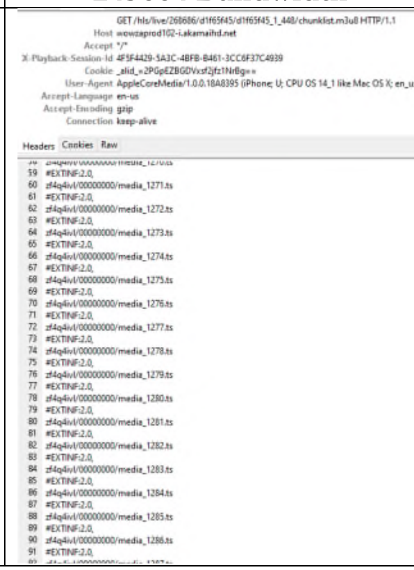
6434112 Bandwidth



403824 Bandwidth



249664 Bandwidth

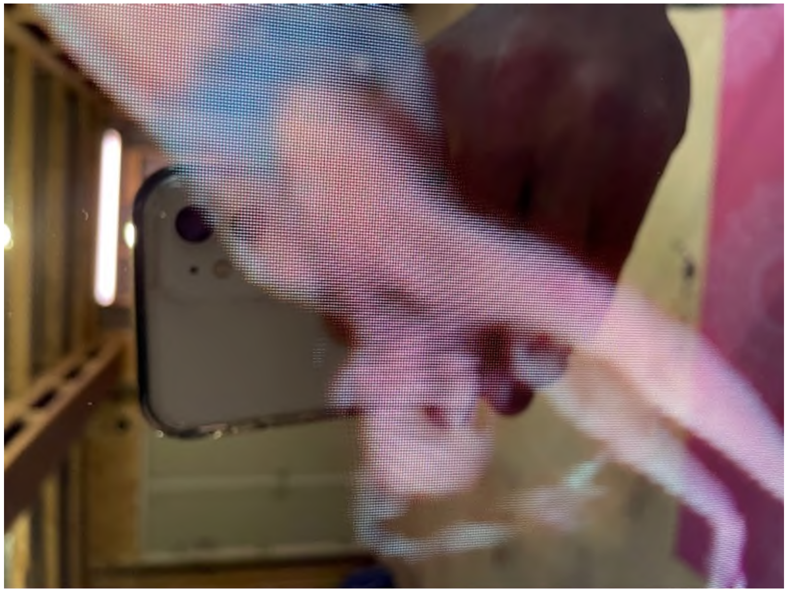


Claim	Claim Limitation	Example Infringement Evidence
		Upon information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application. For example, during a test of the Mirror Devices, a first version, a second version, and a third version of the live video were captured. The first version corresponds to a high-quality stream, the second version corresponds to a medium-quality stream, and the third version corresponds to a low-quality stream. First version: 

USP 10,469,555 to Mirror

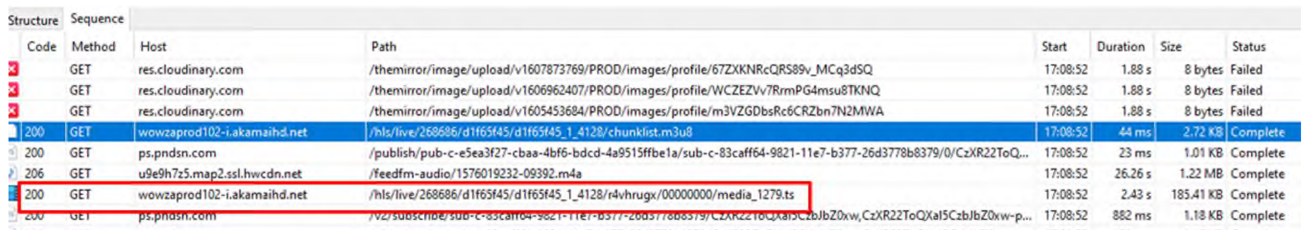
Claim	Claim Limitation	Example Infringement Evidence
		Second version: 

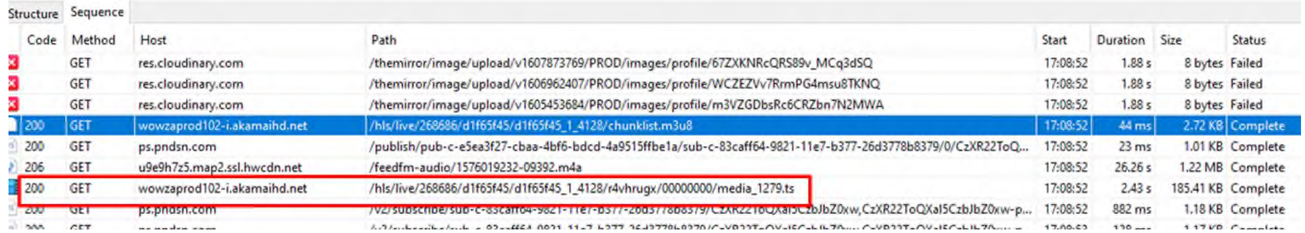
USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Third version: 
	wherein at least one of the low quality stream, medium-quality stream, and high quality stream is encoded at a bit rate of no less than 600 kbps; and	As shown above, "at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bit rate of no less than 600 kbps." At least the high-quality stream (**6326576 Bandwidth**) and one of the medium quality streams (**864048 Bandwidth**) is encoded at a bitrate of not less than 600 kbps as indicated by its "BANDWIDTH" attribute, which signals the upper bound of the overall bitrate for the streamlets in bits per second and is listed at over 6 megabits and 800 kilobits per second.
	wherein the streamlet encoding the same portion of the video in the low quality stream has an equal playback duration as	As shown above, "the streamlet encoding the same portion of the video in the low quality stream has an equal playback duration of the streamlet encoding the same portion of the video in the high quality stream." As discussed above, each of the **6434112 Bandwidth**, the **403824 Bandwidth**, and **249664 Bandwidth** variant playlists includes a first streamlet (e.g., **media_1275.ts** segment). Each of the

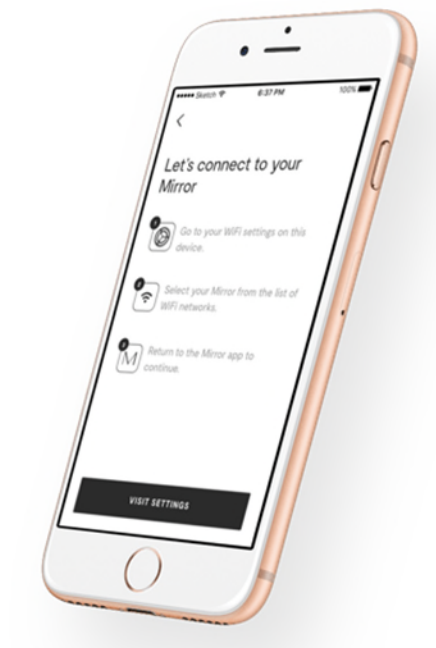
Claim	Claim Limitation	Example Infringement Evidence
	the streamlet encoding the same portion of the video in the high quality stream;	variant "**media_1275.ts**" segments have an "equal playback duration" of 2.0 seconds (as indicated in each line beginning with "#EXTINF") and encode the same portion of the video available in the Mirror Application in different bitrates. Upon information and belief, this is also true for the Mirror Devices as explained above.
	select a specific one of the streams based upon a determination by the client module to select a higher or lower bitrate version of the streams;	The non-transitory machine-readable instructions of the Mirror Application and the Mirror Devices cause the processor to "select a specific one of the streams based upon a determination by the client module to select a higher or lower bitrate version of the streams." Based upon, at least in part, a determination of the available bandwidth, the Mirror Application and Mirror Devices may determine to "select a higher or lower bitrate version of the streams" and thereby "select a specific one of" the low-quality stream (e.g., the **249664 Bandwidth** stream), the medium-quality stream (e.g., the **403824 Bandwidth** stream), and the high-quality stream (e.g., the **6434112 Bandwidth** stream). As part of the testing, the Mirror Application was connected to the Internet through the Charles Proxy application. For the instant test, the Mirror Application selects the **403824 Bandwidth** stream as indicated by its request for a **403824 Bandwidth** playlist (*see* GET request for d1f65f45_1_1728/chunklist.m3u8) and subsequent request for the **403824 Bandwidth** version of the "media_1277.ts" file. When the available bandwidth was reduced during the test, the Mirror Application subsequently determined to and selected a different, lower bandwidth version of the stream. Below is an excerpt of the Charles Proxy application "Sequence" listing showing the same.

Method	Host	Path
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8
..		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
…		

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
<table><tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts</td></tr></table>
Upon information and belief, for at least the reasons stated above, the Mirror Devices and the Mirror Application operate in the same or substantially the same way.
</td></tr>
<tr><td></td><td>place a streamlet request to the server over the one or more network connections for the selected stream;</td><td>
The non-transitory machine-readable instructions of the Mirror Application and Mirror Devices cause the processor to “place a streamlet request to the server over the one or more network connections for the selected stream.”
For the instant test, the Mirror Application requests the 6434112 Bandwidth version of the “media_1279.ts” file. Below is an excerpt of the Charles “Sequence” listing showing the same.

Structure | Sequence
<table>
<tr><th>Code</th><th>Method</th><th>Host</th><th>Path</th><th>Start</th><th>Duration</th><th>Size</th><th>Status</th></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1607873769/PROD/images/profile/67ZXKNRcQRS89v_MCq3dSQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1606962407/PROD/images/profile/WCZEZVv7RrmPG4msu8TKNQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1605453684/PROD/images/profile/m3VZGDbsRc6CRZbn7N2MWA</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8</td><td>17:08:52</td><td>44 ms</td><td>2.72 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>/publish/pub-c-e5ea3f27-cbaa-4bf6-bdcd-4a9515ffbe1a/sub-c-83caff64-9821-11e7-b377-26d3778b8379/0/CzXR22ToQ...</td><td>17:08:52</td><td>23 ms</td><td>1.01 KB</td><td>Complete</td></tr>
<tr><td>206</td><td>GET</td><td>u9e9h7z5.map2.ssl.hwcdn.net</td><td>/feedfm-audio/1576019232-09392.m4a</td><td>17:08:52</td><td>26.26 s</td><td>1.22 MB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/r4vhrugx/00000000/media_1279.ts</td><td>17:08:52</td><td>2.43 s</td><td>185.41 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>/v2/subscribe/sub-c-83caff64-9821-11e7-b377-26d3778b8379/CzXR22ToQXal5CzbJbZ0xw,CzXR22ToQXal5CzbJbZ0xw-p...</td><td>17:08:52</td><td>882 ms</td><td>1.18 KB</td><td>Complete</td></tr>
</table>
Upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.
</td></tr>
<tr><td></td><td>receive the requested streamlets from the server via the one or more network connections; and</td><td>
The non-transitory machine-readable Mirror Application and Mirror Devices cause the processor to “receive the requested streamlets from the server via the one or more network connections.”
For the instant test, the Mirror Application receives the 6434112 Bandwidth version of the “media_1279.ts” file. Below is an excerpt of the Charles “Sequence” listing showing the request is “complete.”
</td></tr>
</table>

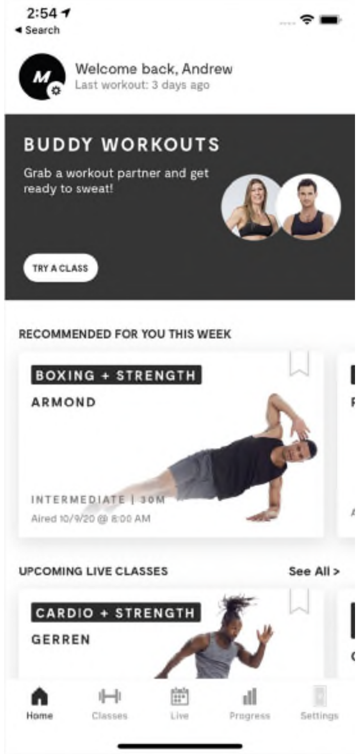
Claim	Claim Limitation	Example Infringement Evidence
		 As shown above, the Mirror Application will continue to receive successive streamlets. Upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.
	provide the received streamlets for playback of the video.	The non-transitory machine-readable instructions of the Mirror Application and Mirror Devices cause the processor to "provide the received first streamlet for playback of the live event video." As described above, the Mirror Application provides, or displays, the received **6434112 Bandwidth** version of the "01279.ts" segment corresponding to the live event video on the screen of the end user station executing the Mirror Application or Mirror Device. In at least this way, upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.
26	A content player device to stream a video over a network from a server for playback of the video, the content player device comprising:	The Mirror Application is software that permits an end user station having a "content player device to stream a video over a network from a server for playback of the video." The Mirror Application is executable by end user stations that have a content player device and the Mirror Application obtains streams of a selected video for playback of the video. The streams are obtained over a network. The exemplary images in this chart of the Mirror Application are from the Mirror Application running on an Apple iPhone XS (Mirror's iOS Application). In addition, the Mirror Application is available to run on other devices, except as otherwise noted, each of these devices is an end user station having a "content player device." **MIRROR DIGITAL OVERVIEW** Access all of MIRROR's Live and On Demand Classes 24/7 from your phone, tablet, and smart TV devices. This feature is included for free with the MIRROR Membership and access is available immediately (whether you already have your Mirror or are awaiting delivery). Please note, it is not currently possible to purchase MIRROR Digital if you do not own a Mirror.

Claim	Claim Limitation	Example Infringement Evidence
		https://mirror.kustomer.help/en_us/mirror-digital-overview-B1pnVd8UU **STREAMING MIRROR DIGITAL TO TV** You can stream classes from the MIRROR App to your television using below devices. The casting devices listed below have been tested by the MIRROR team and are compatible with MIRROR Digital. While additional devices may be compatible, MIRROR cannot guarantee functionality until the device has been thoroughly tested. We will continue to add to this list as MIRROR Digital develops! **iOS Casting Compatible Devices:** ● Apple TV (all models except the first generation) ● Airport Express ● Chromecast ● There are third-party apps you can use to stream to other devices, but they may not be supported in the MIRROR App **Android Casting Compatible Devices:** ● Chromecast ● Android TV ● Google TV ● Samsung TV https://mirror.kustomer.help/en_us/streaming-mirror-digital-to-tv-rJNcuu8I8

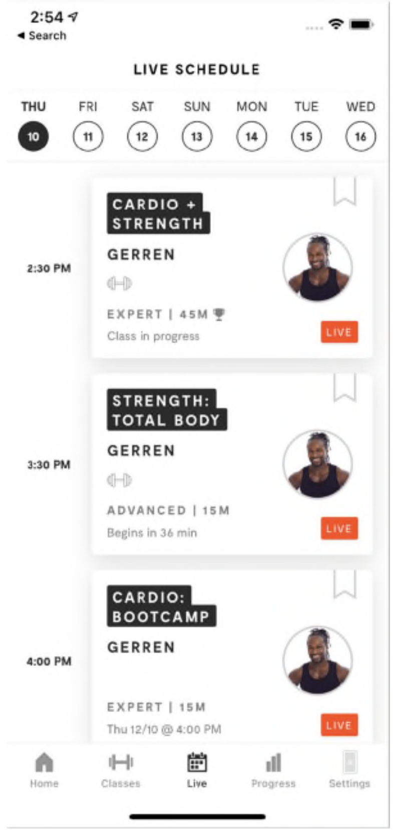
Claim	Claim Limitation	Example Infringement Evidence
		GET THE MIRROR APP To get started setting up your Mirror, you need to download the MIRROR app from the Apple App Store or Google Play Store. The app will take you through everything you need to know. Available on the App Store Get it on Google play Need help? Email us at hello@mirror.co  https://www.mirror.co/app. **MIRROR APP** The MIRROR App allows you to access and customize the Mirror experience. The MIRROR App is available for both iOS and Android! • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later.

Claim	Claim Limitation	Example Infringement Evidence
		https://mirror.kustomer.help/en_us/mirror-app--S1dDC_tYm.  Source: Apple App Store When launched, the Mirror Application displays a main menu:

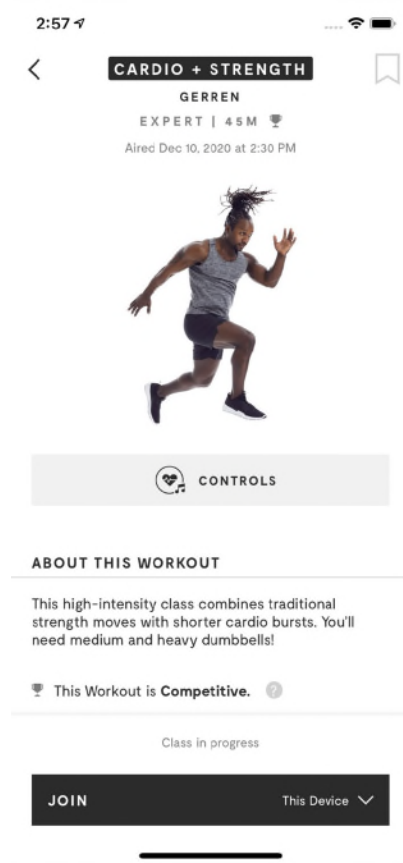
USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application The main menu of the Mirror Application displays classes that are each “a video.” The “Upcoming Live Classes” section of the Mirror Application main menu displays a preview of ongoing and upcoming live video programming.

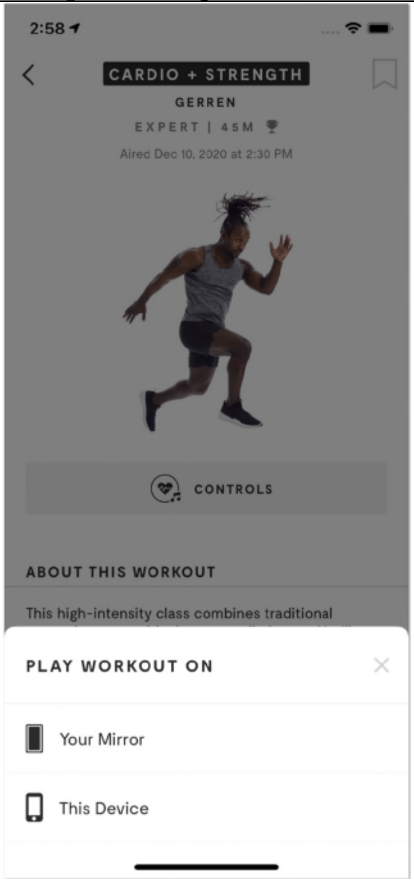
USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting an ongoing live class from the list causes the Mirror Application to display more details regarding the class and provides the user with the option to join the class.

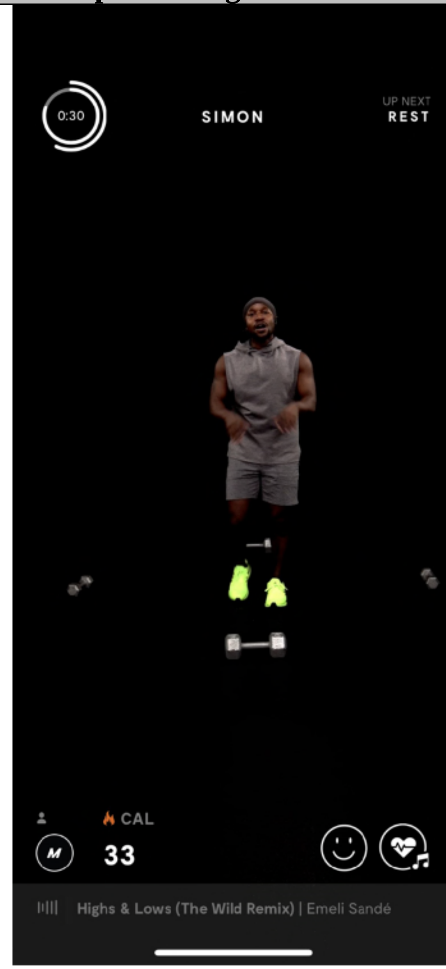
USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting the "Join" button for an ongoing live video causes the Mirror Application to provide options to stream the class to a variety of end user stations having content players over the Internet, including the iOS device that the user is using or the separate Mirror device.

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting “Your Mirror” causes the selected video and other materials to be streamed on the user’s Mirror device, which includes a content player device that is connected to the Internet.

Claim	Claim Limitation	Example Infringement Evidence
		 Alternatively, selecting “This Device” causes the selected workout video and other materials to be streamed on the user’s iOS device:

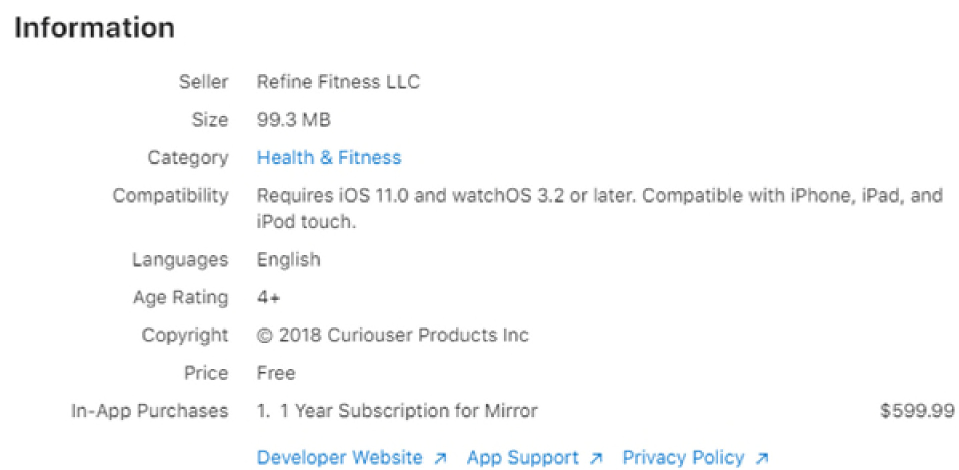
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application As shown above, Mirror Devices include content player devices to stream a video over a network for playback of the video. The Mirror Devices obtain streams of a selected video program for playback. The streams are obtained over a network.

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
	a processor;	A device running the Mirror Application includes at least one "processor" to execute the Mirror Application and stream the live event video. The devices that are compatible with the Mirror Application, each include one or more processors. **MIRROR DIGITAL COMPATIBLE DEVICES** The MIRROR App is available for the iPhone, iPad, Android phones, and Android tablets. MIRROR Digital can be casted to your smart TV using these devices. • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-digital-compatible-devices-HklDdOU8U. For example, Mirror requires users to provide a user device such as an iPhone that includes a processor to execute the Mirror Application. **Information** Seller: Refine Fitness LLC Size: 99.3 MB Category: Health & Fitness Compatibility: Requires iOS 11.0 and watchOS 3.2 or later. Compatible with iPhone, iPad, and iPod touch. Languages: English Age Rating: 4+ Copyright: © 2018 Curiouser Products Inc Price: Free In-App Purchases: 1. 1 Year Subscription for Mirror $599.99 Developer Website ↗ App Support ↗ Privacy Policy ↗ https://apps.apple.com/us/app/mirror-workout-companion/id1153358600.

Claim	Claim Limitation	Example Infringement Evidence
		The Mirror Devices also include a processor. **HARDWARE** FRAME — Carbon steel frame; Mineral bronze powder coated DISPLAY — 40" full HD 1080p display, with 178° wide viewing angle TECHNOLOGY — Quad core processor SOUND — 2 x 10 watt high-fidelity stereo speakers; Embedded omnidirectional microphone CAMERA — 5 megapixel front-facing camera POWER — 1 ft and 6 ft right angle UL certified cables https://www.mirror.co/shop/mirror.
	a digital processing apparatus memory device comprising non-transitory	The device executes the Mirror Application from "a digital processing apparatus memory device comprising non-transitory machine-readable instructions." The instructions include at least the executable instructions for the Mirror Application and its features. Mirror requires users of the

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
	machine-readable instructions that, when executed, cause the processor to:	Mirror Application to provide a device with a digital processing apparatus memory device to store the instructions. For example, Mirror requires users to provide at least 99.3 MB of storage on a digital processing apparatus memory device of the end user station for storing the Mirror Application.  **Information** Seller: Refine Fitness LLC Size: 99.3 MB Category: Health & Fitness Compatibility: Requires iOS 11.0 and watchOS 3.2 or later. Compatible with iPhone, iPad, and iPod touch. Languages: English Age Rating: 4+ Copyright: © 2018 Curiouser Products Inc Price: Free In-App Purchases: 1. 1 Year Subscription for Mirror $599.99 Developer Website ↗ App Support ↗ Privacy Policy ↗ https://apps.apple.com/us/app/mirror-workout-companion/id1153358600. The Mirror Devices also include "a digital processing apparatus memory device comprising non-transitory machine-readable instructions." For example, the on-board quad core processor requires memory containing non-transitory machine-readable instructions in order to process and display streaming fitness classes.

Claim	Claim Limitation	Example Infringement Evidence
		HARDWARE FRAME: Carbon steel frame Mineral bronze powder coated DISPLAY: 40" full HD 1080p display, with 178° wide viewing angle TECHNOLOGY: Quad core processor SOUND: 2 x 10 watt high-fidelity stereo speakers Embedded omnidirectional microphone CAMERA: 5 megapixel front-facing camera POWER: 1 ft and 6 ft right angle UL certified cables https://www.mirror.co/shop/mirror.
	establish one or more network connections between the client module and the server, wherein the server is configured	As shown below, the non-transitory machine-readable instructions of the Mirror Application and Mirror Devices, when executed, cause the processor(s) to "establish one or more network connections between the client module and the server" that is "configured to access at least one of a plurality of groups of streamlets." The "segments" discussed herein are "streamlets." The Mirror Application requires an internet connection.

Claim	Claim Limitation	Example Infringement Evidence
	to access at least one of a plurality of groups of streamlets;	**PRELOAD CLASSES ON MIRROR DIGITAL** You currently cannot preload classes on the MIRROR Digital, however this feature is coming soon! If you are not able to connect WiFi or are in a tough WiFi environment, you can always use cellular data to stream classes on the MIRROR App. Please consult your cell phone provider for questions about data usage and your plan. https://mirror.kustomer.help/en_us/preload-classes-on-mirror-app-H12XPdUUL. To stream a video, such as that shown above, the Mirror Application requests a stream of a selected live event video via a network connection. A user may select to stream a live event video by selecting the Join button, as shown above. When the Mirror Application accesses a selected live program, it requests and receives a playlist file that shows the available versions of the program at different resolutions. For the following test, a live event video was selected. In the test, an iPhone 11 running the Mirror Application makes an HTTPS GET request for a master playlist named "**playlist.m3u8**" that specifies the available streams and provides links to the playlists for those streams. The following master playlist named "**playlist.m3u8**" is returned.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080
../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720
../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480
../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360
../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288
../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180
../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8

Filename: playlist.m3u8.

This is a master playlist file according to the HLS specification.[2] The playlist shows six versions of the stream, denoted by each #EXT-X-STREAM-INF tag at the following bandwidths:
• 6434112 (referred to herein as “6434112 Bandwidth”)
• 864048 (referred to herein as “864048 Bandwidth”)
• 403824 (referred to herein as “403824 Bandwidth”)
• 367728 (referred to herein as “367728 Bandwidth”)
• 312832 (referred to herein as “312832 Bandwidth”)
• 249664 (referred to herein as “249664 Bandwidth”)

For each of these versions, the master playlist provides a link to a playlist file for the specified version of the selected video at a particular bandwidth and resolution, which is called a “variant” in HLS.

The Mirror Application initially selects the 6434112 Bandwidth (1080p – high bandwidth) version of the video and makes a request for the corresponding variant playlist file named</td></tr>
</table>

[2] RFC 8216 (HLS Live Streaming), Section 4.3.4 (Master Playlist Tags)

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		"**chunklist.m3u8**." That file is returned with the following contents (a portion of which is shown below). #EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1232 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:08.356Z #EXTINF:2.0, r4vhrugx/00000000/media_1232.ts #EXTINF:2.0, r4vhrugx/00000000/media_1233.ts #EXTINF:2.0, r4vhrugx/00000000/media_1234.ts #EXTINF:2.0, r4vhrugx/00000000/media_1235.ts #EXTINF:2.0, r4vhrugx/00000000/media_1236.ts #EXTINF:2.0, r4vhrugx/00000000/media_1237.ts #EXTINF:2.0, r4vhrugx/00000000/media_1238.ts #EXTINF:2.0, r4vhrugx/00000000/media_1239.ts #EXTINF:2.0, r4vhrugx/00000000/media_1240.ts #EXTINF:2.0, r4vhrugx/00000000/media_1241.ts #EXTINF:2.0, Path: https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 As noted above, the variant playlist file is an HLS playlist. The variant playlist file identifies a plurality of segments or "streamlets" that are part of the **6434112 Bandwidth** group of streamlets. Each line

Claim	Claim Limitation	Example Infringement Evidence
		in the file "**chunklist.m3u8**" that begins with "#EXTINF" specifies the length of the segments in seconds (2.0). The line below the #EXTINF entry is the relative location of the video file for the segment (e.g., r4vhrugx/0000000/media_1232.ts). The Mirror Application uses HTTPS GET requests to retrieve the segments of the encoded video specified in the file above , which accesses the segments in response to requests from the Mirror Application. The Mirror Application makes a request for a segment media_1232.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected video. As long as the viewer stays on the selected video and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **6434112 Bandwidth** version) . While the example above relates to the **6434112 Bandwidth** group of streamlets, other groups of streamlets are also available. If the available bandwidth for the network connection decreases, for example, the Mirror Application will continue to request segments to continue streaming the selected video, but at one of the lower bandwidths. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variant playlist file for the **403824 Bandwidth** named "chunklist.m3u8." That file is returned with the following contents showing a portion of the **403824 Bandwidth** group of segments for the video being streamed.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1238
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z
#EXTINF:2.0,
fbd862nq/00000000/media_1238.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1239.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1240.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1241.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1242.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1243.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1244.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1245.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1246.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1247.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1248.ts
```

Filename: **chunklist.m3u8**

The Mirror Application then makes the request for media_1238.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected video at the **403824 Bandwidth**. As long as the viewer stays on the selected video and the bandwidth is adequate, the Mirror Application will continue to request and
</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		receive playlists corresponding to the current, chosen resolution (in this case, the **403824 Bandwidth** version). As the bandwidth is further constrained, the Mirror Application makes another request for a lower quality stream of the selected video. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variable playlist file for the **249664 Bandwidth** named “chunklist.m3u8.” That file is returned with the following contents showing a portion of the **249664 Bandwidth** group of segments for the selected video being streamed. #EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1241 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:26.354Z #EXTINF:2.0, zf4q4ivl/00000000/media_1241.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1242.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1243.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1244.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1245.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1246.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1247.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1248.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1249.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1250.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1251.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1252.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1253.ts

Claim	Claim Limitation	Example Infringement Evidence
		File: **chunklist.m3u8** The Mirror Application then makes the request for media_1281.ts, the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected video at the **249664 Bandwidth**. As long as the viewer stays on the selected video and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **249664 Bandwidth** version). A portion of a subsequently retrieved playlist's contents are shown below.

USP 10,469,555 to Mirror

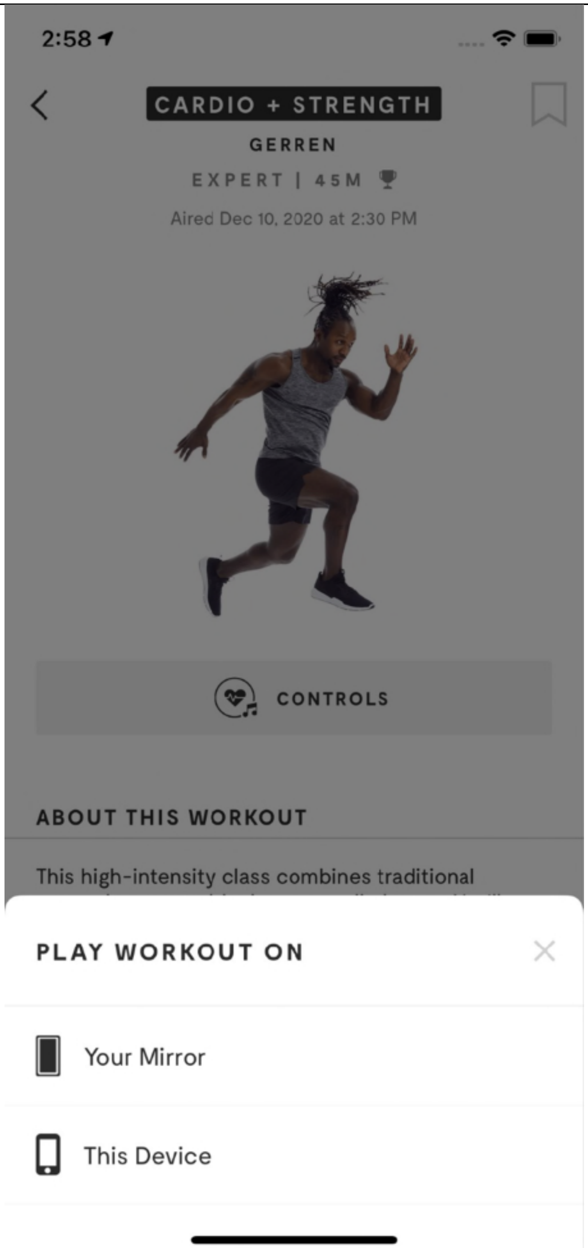
Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1245 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:34.354Z #EXTINF:2.0, zf4q4ivl/00000000/media_1245.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1246.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1247.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1248.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1249.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1250.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1251.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1252.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1253.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1254.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1255.ts #EXTINF:2.0, Filename: **chunklist.m3u8** The subsequently retrieved playlist includes additional video segments that were not included in the previous playlist file, for example: "**media_1254.ts**" and "**media_1255.ts.**" The Mirror Application continues to request, receive, and playback successive segments of the video to stream the video. The Mirror Devices also require an internet connection.

USP 10,469,555 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
CONNECTION
INTERNET — Dual-band 802.11 A/B/G/N Wi-Fi
APP — Controlled by iOS or Android companion app
HEART RATE — Syncs with Bluetooth™ heart rate monitors, Apple Watches, and Android Wear OS Watches
AUDIO — Pairs with Bluetooth™ speakers and headphones
https://www.mirror.co/shop/mirror

To stream a video, such as that shown above, the Mirror Device requests a stream of a selected live event video via a network connection. The iOS application provides the interface for interacting with a Mirror Device (i.e., selecting a live or on demand class to stream). After selecting a class, the user selects whether to use the iOS device or Mirror Device to view the content (i.e., stream and participate in the workout).
</td></tr>
</table>

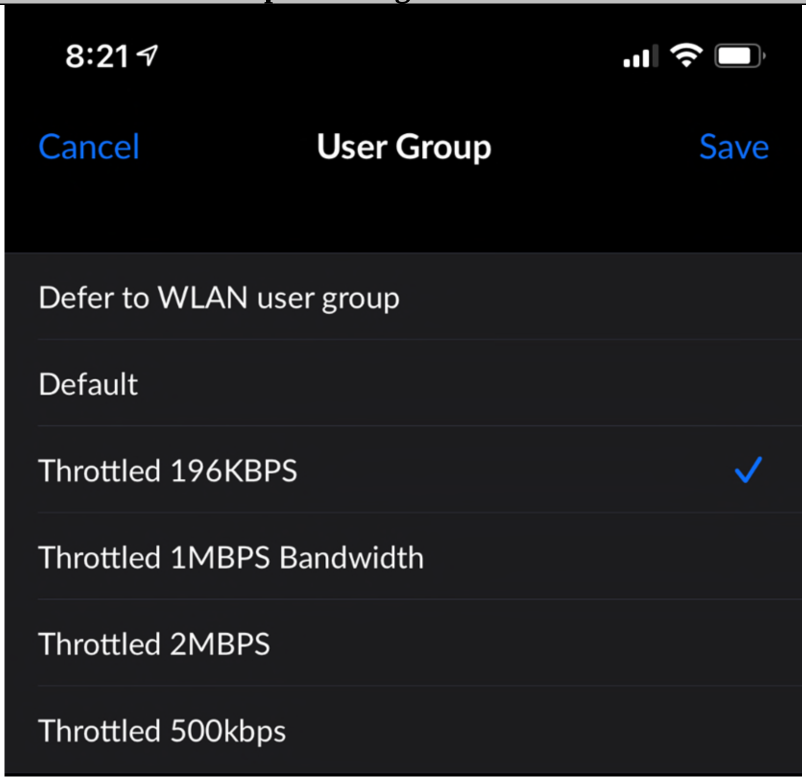
USP 10,469,555 to Mirror



Claim	Claim Limitation	Example Infringement Evidence
		<u>Source: Mirror iOS Application</u> Selecting "Your Mirror" causes the Mirror Device to initiate streaming requests:  For the following test, a live video was selected. Based on the test, and upon information and belief, the Mirror Devices make the same or substantially the same requests to and retrievals as shown above with respect to the Mirror Application. For example, when the Mirror Device(s) accesses a selected video, the Mirror Device(s) initially selects a first bandwidth version of the stream, makes a request for the segments of the group corresponding to the selected bandwidth version of the live event program, receives segments from the group corresponding to the selected bandwidth version, and then plays the requested and received segments on the Mirror Device content player as shown below.

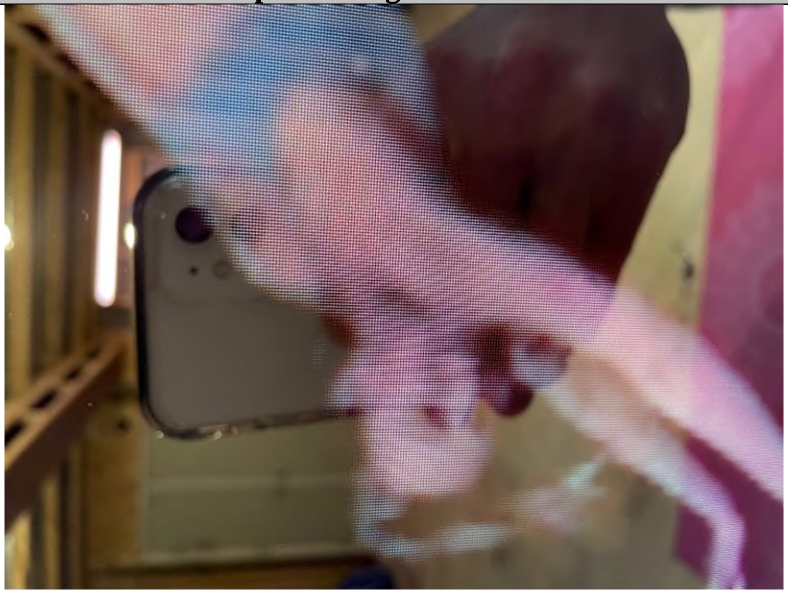
USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Other groups of streamlets are also available. For example, for the current test, bandwidth for the Mirror Device was constrained to 196Kbps, which caused the Mirror Device to display a "buffering" message while requesting and receiving a corresponding playlist and streamlets for a second bandwidth version of the video as shown below.

Claim	Claim Limitation	Example Infringement Evidence
		

USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 The image resolution for the second bandwidth streamlet requested is noticeably lower quality as indicated by the pixelated edges of the instructor's body, as shown below.

Claim	Claim Limitation	Example Infringement Evidence
		
	wherein the video is encoded at a plurality of different bitrates to create a plurality of streams including at least a low quality stream and a high quality stream, the low quality stream and the high quality stream each representing the same portion of the video with a streamlets encoded at a different	The “video is encoded at a plurality of different bitrates to create a plurality of streams.” The plurality of different bitrates creates a plurality of streams “including at least a low quality stream, a medium quality stream, and a high quality stream.” And “the low quality stream and the high quality stream each represent[] the same portion of the video with streamlets encoded at a different one of the plurality of different bitrates.” These elements are shown below. As shown in the master playlist file, “**playlist.m3u8**,” the live video available for the selected video program is encoded at 6 different bitrates.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td>one of the plurality of different bitrates; and</td><td>

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080
../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720
../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480
../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360
../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288
../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180
../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
```

Filename: playlist.m3u8.

The six different bitrates correspond to six different video streams of the same content but at varying quality:

- 6434112 (referred to herein as “**6434112 Bandwidth**”)
- 864048 (referred to herein as “**864048 Bandwidth**”)
- 403824 (referred to herein as “**403824 Bandwidth**”)
- 367728 (referred to herein as “**367728 Bandwidth**”)
- 312832 (referred to herein as “**312832 Bandwidth**”)
- 249664 (referred to herein as “**249664 Bandwidth**”)

These different bitrate versions include at least “a low quality stream, a medium quality stream, and a high quality stream.” For example, the **6434112 Bandwidth** version can be considered a high-quality stream, the **403824 Bandwidth** version can be considered a medium-quality stream, and the **249664 Bandwidth** version can be considered a low-quality stream.

As shown herein, each of the high-quality stream (e.g., the **6434112 Bandwidth** stream), the medium-quality stream (e.g., the **403824 Bandwidth** stream), and the low-quality stream (e.g., the
</td></tr>
</table>

USP 10,469,555 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
249664 Bandwidth stream) comprise a group of streamlets that are encoded at the same respective one of the different bitrates. Each variant playlist includes at least two streamlets: a media_1274.ts” segment and a “media_1275.ts” segment. A comparison of the 6434112 Bandwidth, 403823 Bandwidth, and 249664 Bandwidth versions below shows that each playlist includes segments with these file names. On information and belief, playlists for the other three variants also include these segments.

As discussed above, each streamlet corresponds to a portion of the live video available on the live video stream. Notably, each bitrate version of the media_1275.ts segment has a duration of 2.0 seconds (as indicated in each line beginning with “#EXTINF”), “representing the same portion of the video with streamlets encoded at a different one of the plurality of bitrates.”

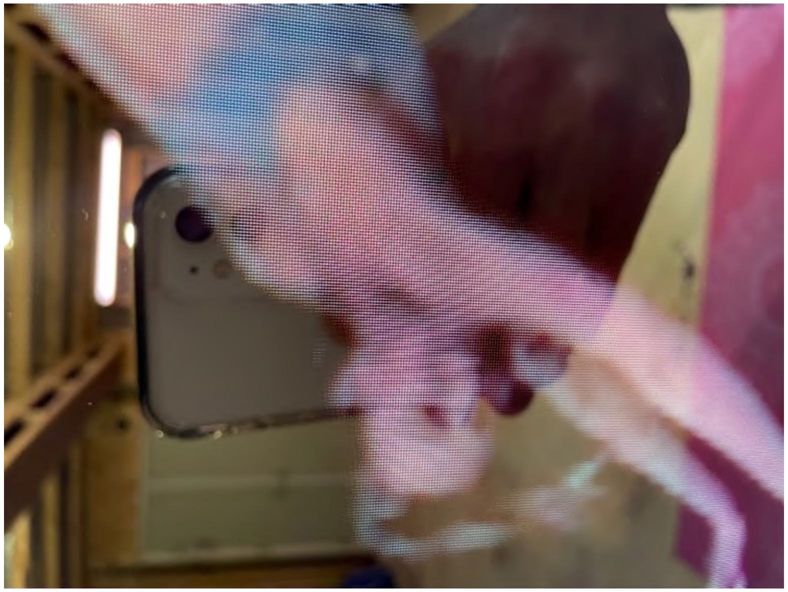
<table>
<tr><th>6434112 Bandwidth</th><th>403824 Bandwidth</th><th>249664 Bandwidth</th></tr>
<tr><td></td><td></td><td></td></tr>
</table>

Upon information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application. For example, during a test of the Mirror Devices, a first version, a second version, and a third version of the live video were captured. The first version corresponds to a high-
</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
		quality stream, the second version corresponds to a medium-quality stream, and the third version corresponds to a low-quality stream. First version: 

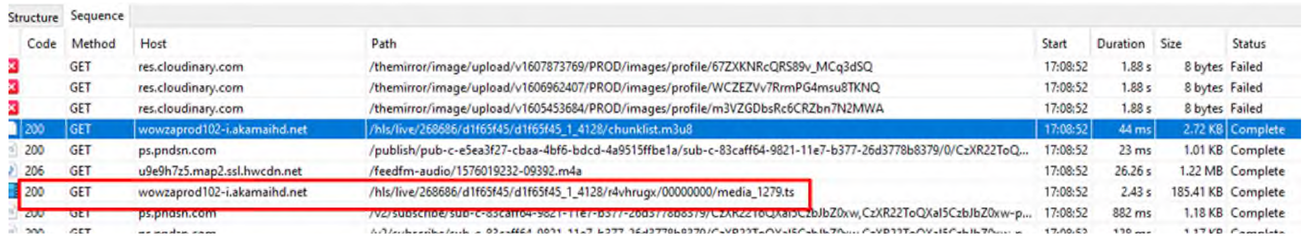
Claim	Claim Limitation	Example Infringement Evidence
		Second version: 

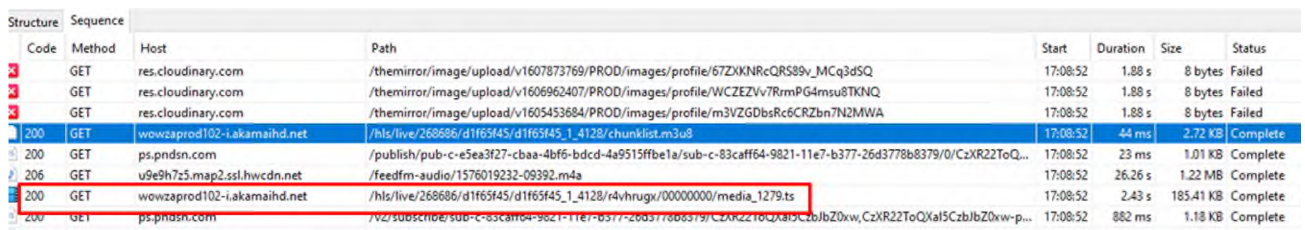
USP 10,469,555 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Third version: 
	wherein the streamlet representing the same portion of the video in the low quality stream and the streamlet representing the same portion of the video in the high quality stream have durations equal to each other;	As shown above, "the streamlet representing the same portion of the video in the low quality stream and the streamlet representing the same portion of the video in the high quality stream have durations equal to each other." As discussed above, each of the **6434112 Bandwidth**, the **403824 Bandwidth**, and **249664 Bandwidth** variant playlists includes a streamlet (e.g., **media_1275.ts** segment). Each of the variant "**media_1275.ts**" segment "have durations equal to each other" of 2.0 seconds (as indicated in each line beginning with "#EXTINF") and encode the same portion of the live video available in the Mirror Application in different bitrates. Upon information and belief, this is also true for the Mirror Devices as explained above.
	select a specific one of the streams based upon a determination	The non-transitory machine-readable instructions of the Mirror Application and the Mirror Devices cause the processor to "select a specific one of the streams based upon a determination by the client module to select a higher or lower bitrate version of the streams."

USP 10,469,555 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td>by the client module to select a higher or lower bitrate version of the streams;</td><td>
Based upon, at least in part, a determination of the available bandwidth, the Mirror Application and Mirror Devices may determine to “select a higher or lower bitrate version of the stream” and thereby “select a specific one of” the low-quality stream (e.g., the 249664 Bandwidth stream), the medium-quality stream (e.g., the 403824 Bandwidth stream), and the high-quality stream (e.g., the 6434112 Bandwidth stream).
As part of the testing, the Mirror Application was connected to the Internet through the Charles Proxy application. For the instant test, the Mirror Application selects the 403824 Bandwidth stream as indicated by its request for a 403824 Bandwidth playlist (see GET request for d1f65f45_1_1728/chunklist.m3u8) and subsequent request for the 403824 Bandwidth version of the “media_1277.ts” file. When the available bandwidth was reduced during the test, the Mirror Application subsequently determined to and selected a different, lower bandwidth version of the stream. Below is an excerpt of the Charles Proxy application “Sequence” listing showing the same.
<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8</td></tr>
<tr><td colspan="3">..</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts</td></tr>
</table>
Upon information and belief, for at least the reasons stated above, the Mirror Devices and the Mirror Application operate in the same or substantially the same way.
</td></tr>
<tr><td></td><td>place a streamlet request to the server over the one or more</td><td>The non-transitory machine-readable instructions of the Mirror Application and Mirror Devices cause the processor to “place a streamlet request to the server over the one or more network connections for the selected stream.”</td></tr>
</table>

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td>network connections for the selected stream;</td><td>For the instant test, the Mirror Application requests the 6434112 Bandwidth version of the “media_1279.ts” file. Below is an excerpt of the Charles “Sequence” listing showing the same.

<table>
<tr><th>Code</th><th>Method</th><th>Host</th><th>Path</th><th>Start</th><th>Duration</th><th>Size</th><th>Status</th></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1607873769/PROD/images/profile/67ZXKNRcQRS89v_MCq3dSQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1606962407/PROD/images/profile/WCZEZVv7RrmPG4msu8TKNQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1605453684/PROD/images/profile/m3VZGDbsRc6CRZbn7N2MWA</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8</td><td>17:08:52</td><td>44 ms</td><td>2.72 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>/publish/pub-c-e5ea3f27-cbaa-4bf6-bdcd-4a9515ffbe1a/sub-c-83caff64-9821-11e7-b377-26d3778b8379/0/CzXR22ToQ...</td><td>17:08:52</td><td>23 ms</td><td>1.01 KB</td><td>Complete</td></tr>
<tr><td>206</td><td>GET</td><td>u9e9h7z5.map2.ssl.hwcdn.net</td><td>/feedfm-audio/1576019232-09392.m4a</td><td>17:08:52</td><td>26.26 s</td><td>1.22 MB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/r4vhrugx/00000000/media_1279.ts</td><td>17:08:52</td><td>2.43 s</td><td>185.41 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>...CzbJbZ0xw,CzXR22ToQXal5CzbJbZ0xw-p...</td><td>17:08:52</td><td>882 ms</td><td>1.18 KB</td><td>Complete</td></tr>
</table>
Upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.</td></tr>
<tr><td></td><td>receive the requested streamlets from the server via the one or more network connections; and</td><td>The non-transitory machine-readable instructions of the Mirror Application and Mirror Devices cause the processor to “receive the requested streamlets from the server via the one or more network connections.”

For the instant test, the Mirror Application receives the 6434112 Bandwidth version of the “media_1279.ts” file. Below is an excerpt of the Charles “Sequence” listing showing the request is “complete.”

<table>
<tr><th>Code</th><th>Method</th><th>Host</th><th>Path</th><th>Start</th><th>Duration</th><th>Size</th><th>Status</th></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1607873769/PROD/images/profile/67ZXKNRcQRS89v_MCq3dSQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1606962407/PROD/images/profile/WCZEZVv7RrmPG4msu8TKNQ</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td></td><td>GET</td><td>res.cloudinary.com</td><td>/themirror/image/upload/v1605453684/PROD/images/profile/m3VZGDbsRc6CRZbn7N2MWA</td><td>17:08:52</td><td>1.88 s</td><td>8 bytes</td><td>Failed</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8</td><td>17:08:52</td><td>44 ms</td><td>2.72 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>/publish/pub-c-e5ea3f27-cbaa-4bf6-bdcd-4a9515ffbe1a/sub-c-83caff64-9821-11e7-b377-26d3778b8379/0/CzXR22ToQ...</td><td>17:08:52</td><td>23 ms</td><td>1.01 KB</td><td>Complete</td></tr>
<tr><td>206</td><td>GET</td><td>u9e9h7z5.map2.ssl.hwcdn.net</td><td>/feedfm-audio/1576019232-09392.m4a</td><td>17:08:52</td><td>26.26 s</td><td>1.22 MB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_4128/r4vhrugx/00000000/media_1279.ts</td><td>17:08:52</td><td>2.43 s</td><td>185.41 KB</td><td>Complete</td></tr>
<tr><td>200</td><td>GET</td><td>ps.pndsn.com</td><td>...CzbJbZ0xw,CzXR22ToQXal5CzbJbZ0xw-p...</td><td>17:08:52</td><td>882 ms</td><td>1.18 KB</td><td>Complete</td></tr>
</table>
As shown above, the Mirror Application will continue to receive successive streamlets. Upon information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.</td></tr>
</table>

Claim	Claim Limitation	Example Infringement Evidence
	provide the received streamlets for playback of the video.	The non-transitory machine-readable instructions of the Mirror Application and Mirror Devices cause the processor to "provide the received streamlets for playback of the video." As described above, the Mirror Application provides, or displays, the received **6434112 Bandwidth** version of the "01286.ts" segment corresponding to the video on the screen of the device. In at least this way, upon information and belief, and as explained above, the Mirror Devices operate in the same or substantially the same way as the Mirror Application.

EXHIBIT D

US010757156B2

(12) **United States Patent**
Major et al.

(10) **Patent No.: US 10,757,156 B2**
(45) **Date of Patent: *Aug. 25, 2020**

(54) **APPARATUS, SYSTEM, AND METHOD FOR ADAPTIVE-RATE SHIFTING OF STREAMING CONTENT**

(71) Applicant: **DISH Technologies L.L.C.**, Englewood, CO (US)

(72) Inventors: **Robert Drew Major**, Orem, UT (US); **Mark B. Hurst**, Cedar Hills, UT (US)

(73) Assignee: **DISH Technologies L.L.C.**, Englewood, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **16/291,343**

(22) Filed: **Mar. 4, 2019**

(65) **Prior Publication Data**

US 2019/0199768 A1 Jun. 27, 2019

Related U.S. Application Data

(63) Continuation of application No. 15/207,172, filed on Jul. 11, 2016, now Pat. No. 10,225,304, which is a (Continued)

(51) **Int. Cl.**
H04L 29/06 (2006.01)
H04N 21/258 (2011.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04L 65/601*** (2013.01); ***H04L 43/16*** (2013.01); ***H04L 47/25*** (2013.01); ***H04L 65/60*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC H04N 19/124; H04N 19/132; H04N 19/59; H04N 19/61; H04N 19/12; H04N 19/172;
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

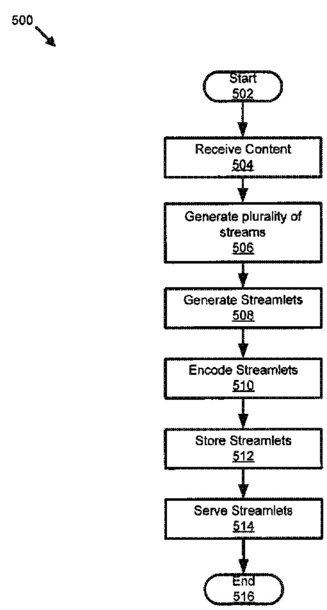
5,953,506 A * 9/1999 Kalra G06T 3/4092 345/428
6,880,017 B1 * 4/2005 Marce H04L 69/32 709/232
(Continued)

Primary Examiner — Ninos Donabed
(74) *Attorney, Agent, or Firm* — Lorenz & Kopf LLP

(57) **ABSTRACT**

An apparatus for adaptive-rate shifting of streaming content includes an agent controller module configured to simultaneously request at least portions of a plurality of streamlets. The agent controller module is further configured to continuously monitor streamlet requests and subsequent responses, and accordingly request higher or lower quality streamlets. A staging module is configured to stage the streamlets and arrange the streamlets for playback on a content player. A system includes a data communications network, a content server coupled to the data communications network and having a content module configured to process content and generate a plurality of high and low quality streams, and the apparatus. A method includes simultaneously requesting at least portions of a plurality of streamlets, continuously monitoring streamlet requests and subsequent responses, and accordingly requesting higher or lower quality streamlets, and staging the streamlets and arranging the streamlets for playback on a content player.

18 Claims, 7 Drawing Sheets



Related U.S. Application Data

continuation of application No. 14/516,303, filed on Oct. 16, 2014, now Pat. No. 9,407,564, which is a continuation of application No. 11/116,783, filed on Apr. 28, 2005, now Pat. No. 8,868,772.

(60) Provisional application No. 60/566,831, filed on Apr. 30, 2004.

(51) **Int. Cl.**

H04N 21/2662	(2011.01)
H04N 21/643	(2011.01)
H04N 21/647	(2011.01)
H04N 21/845	(2011.01)
H04N 21/84	(2011.01)
H04L 12/825	(2013.01)
H04L 12/26	(2006.01)
H04L 29/08	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 67/02*** (2013.01); ***H04L 69/16*** (2013.01); ***H04N 21/25808*** (2013.01); ***H04N 21/2662*** (2013.01); ***H04N 21/643*** (2013.01); ***H04N 21/64769*** (2013.01); ***H04N 21/64792*** (2013.01); ***H04N 21/84*** (2013.01); ***H04N 21/845*** (2013.01)

(58) **Field of Classification Search**

CPC .. H04N 19/187; H04N 19/192; H04N 11/042; H04N 19/00; H04N 19/117; H04N 19/30; H04L 41/145; H04L 43/04; H04L 43/50; G06F 11/3409; G06F 11/348; G06F 16/739; G06F 16/7834; G06F 16/7844

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,257,407	B2 *	8/2007	Dacosta	H04W 28/20	370/468
7,308,487	B1	12/2007	Dansie et al.		
7,324,523	B2 *	1/2008	Dacosta	H04L 41/0896	370/345
7,363,228	B2 *	4/2008	Wyss	G10L 15/30	379/88.03
7,526,565	B2 *	4/2009	Amini	H04L 29/06027	709/231
7,567,746	B2 *	7/2009	Saeki	G11B 27/10	386/239
8,321,584	B2 *	11/2012	Dobbins	H04L 12/14	709/206
8,667,158	B2 *	3/2014	Jin	H04N 21/23418	709/231
2002/0103938	A1 *	8/2002	Brooks	H03M 7/30	709/247
2002/0144276	A1 *	10/2002	Radford	H04N 7/17318	725/87
2002/0159457	A1 *	10/2002	Zhang	H04M 11/062	370/391
2003/0055995	A1 *	3/2003	Ala-Honkola	H04N 21/23406	709/231
2004/0073934	A1 *	4/2004	Deshpande	H04N 7/17318	725/87
2004/0190528	A1 *	9/2004	Dacosta	H04L 41/0896	370/395.41
2004/0192322	A1 *	9/2004	Dacosta	H04W 28/20	455/452.1
2004/0196842	A1 *	10/2004	Dobbins	H04L 12/2856	370/389
2004/0199472	A1 *	10/2004	Dobbins	G06Q 20/123	705/50
2004/0199604	A1 *	10/2004	Dobbins	H04L 12/2856	709/217
2004/0199667	A1 *	10/2004	Dobbins	H04L 12/14	709/240
2004/0210948	A1 *	10/2004	Jin	H04N 21/2312	725/145
2006/0168524	A1 *	7/2006	Saeki	G11B 27/10	715/728
2007/0008884	A1 *	1/2007	Tang	H04L 29/06	370/230
2007/0174471	A1 *	7/2007	Van Rossum	H04L 29/06	709/229

* cited by examiner

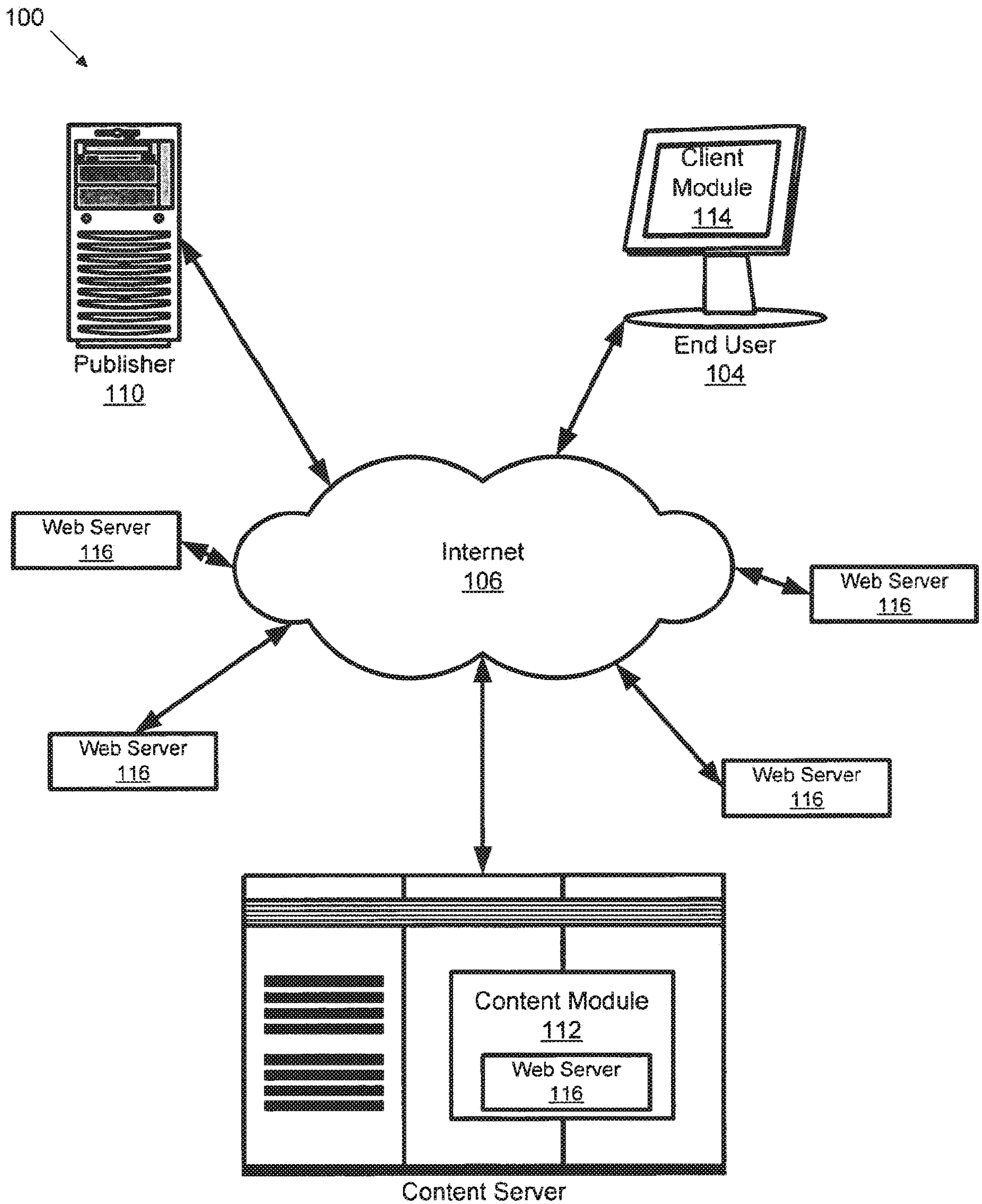

100
Client Module 114
Publisher 110
End User 104
Web Server 116
Internet 106
Web Server 116
Web Server 116
Web Server 116
Content Module 112
Web Server 116
Content Server

FIG. 1

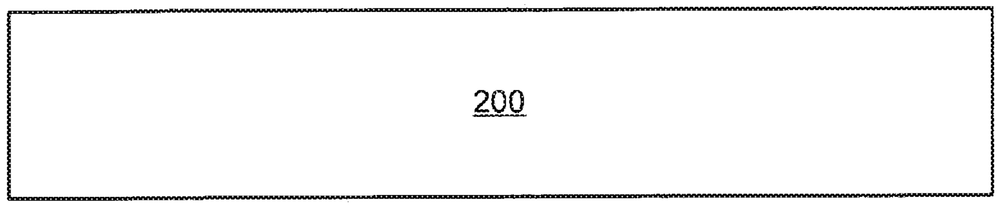


FIG. 2a

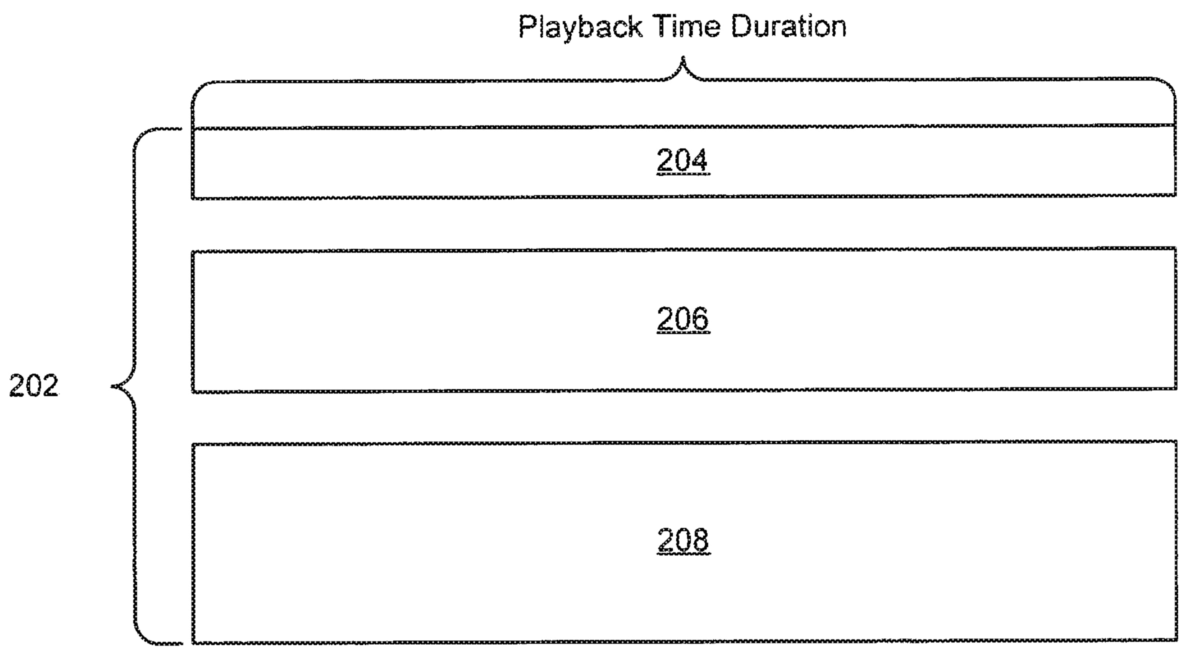


FIG. 2b

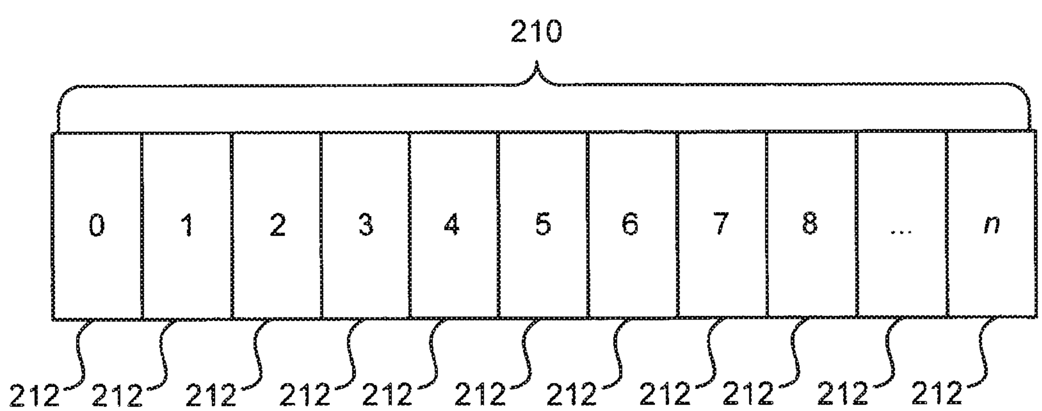


FIG. 2c

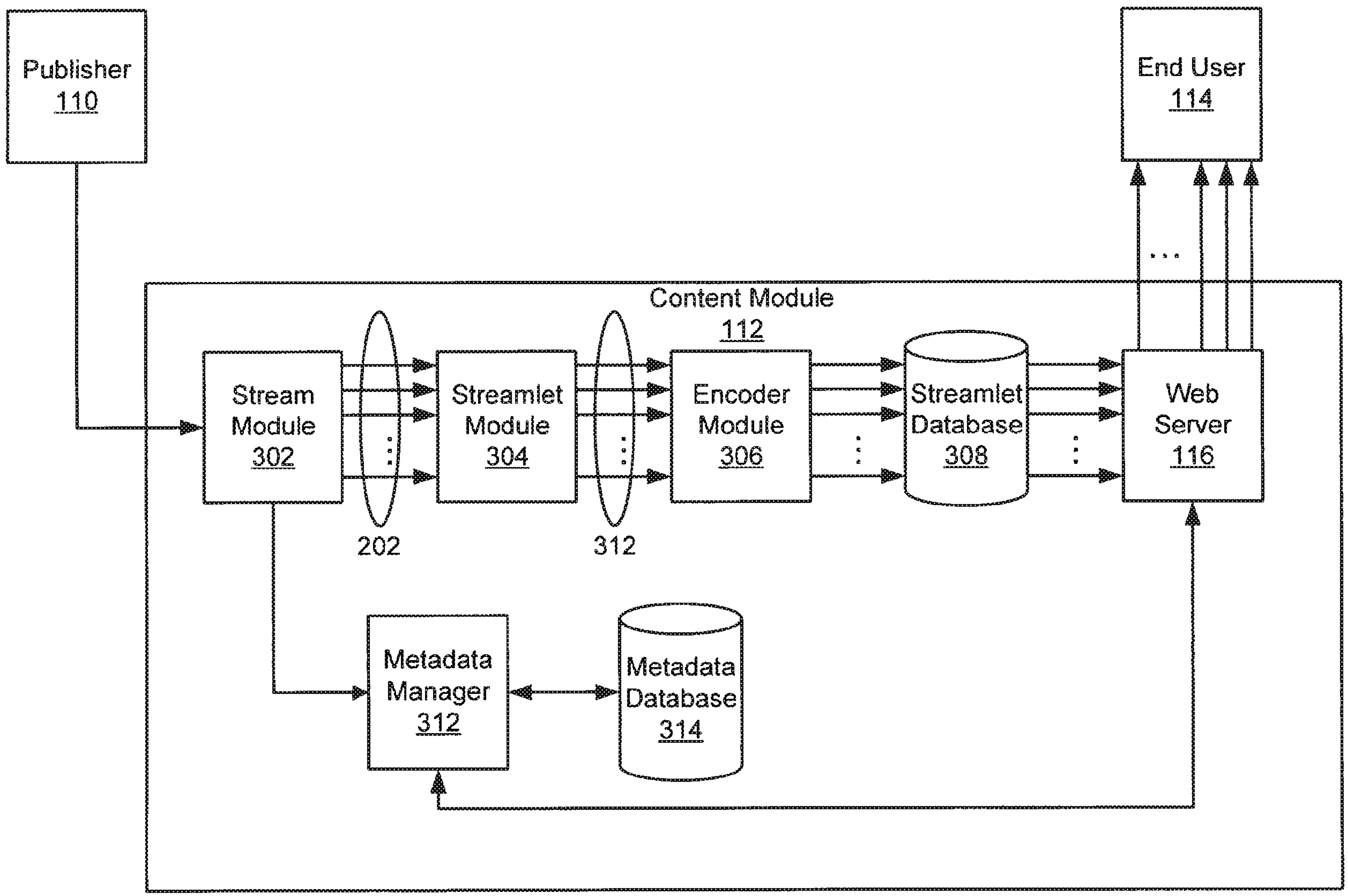

Publisher
110
End User
114
Content Module
112
Stream
Module
302
Streamlet
Module
304
Encoder
Module
306
Streamlet
Database
308
Web
Server
116
202
312
Metadata
Manager
312
Metadata
Database
314

FIG. 3

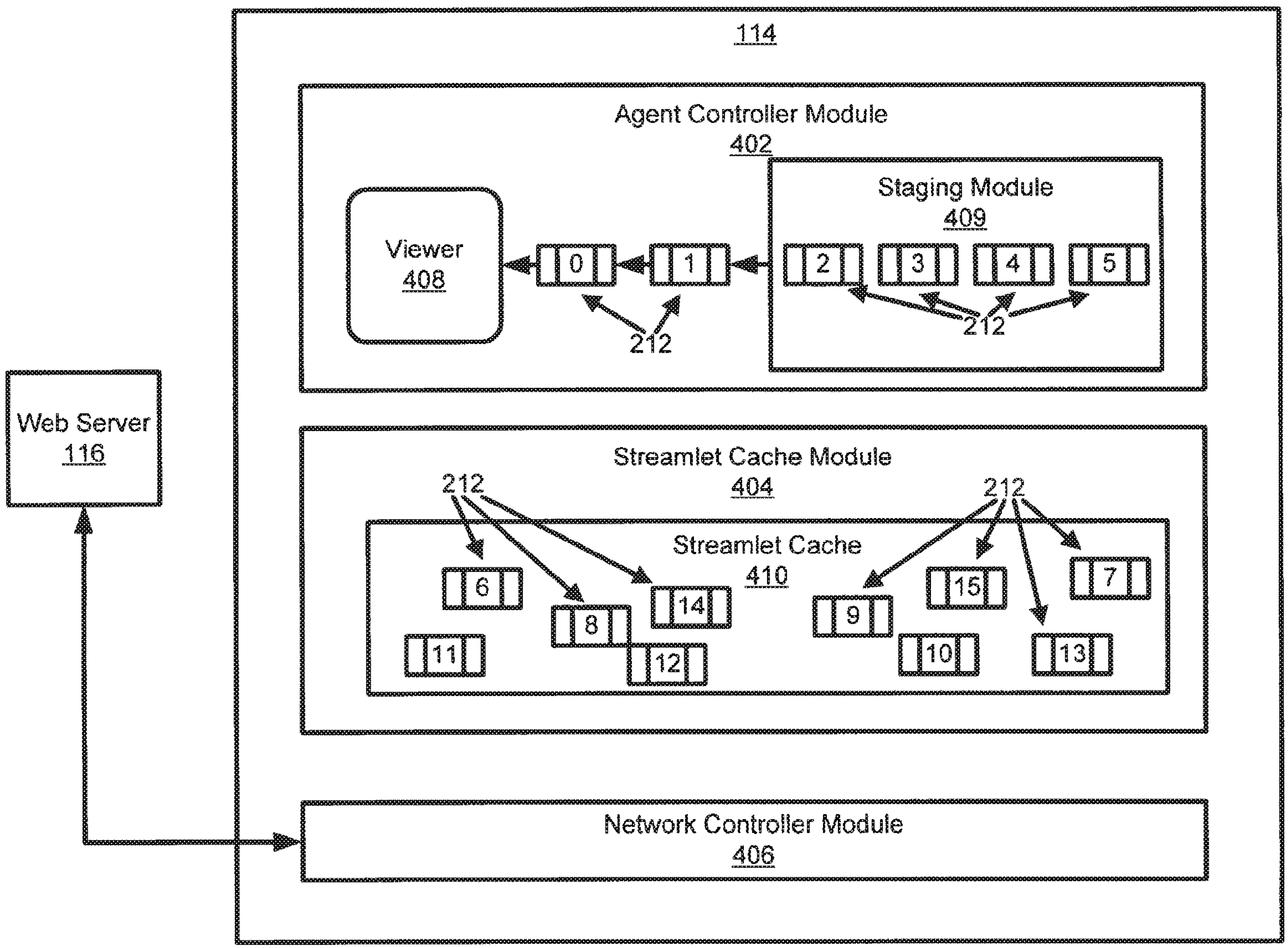

114
Agent Controller Module
402
Viewer
408
0
1
212
Staging Module
409
2
3
4
5
212
Web Server
116
Streamlet Cache Module
404
212
212
Streamlet Cache
410
6
8
14
11
12
9
15
7
10
13
Network Controller Module
406

FIG. 4

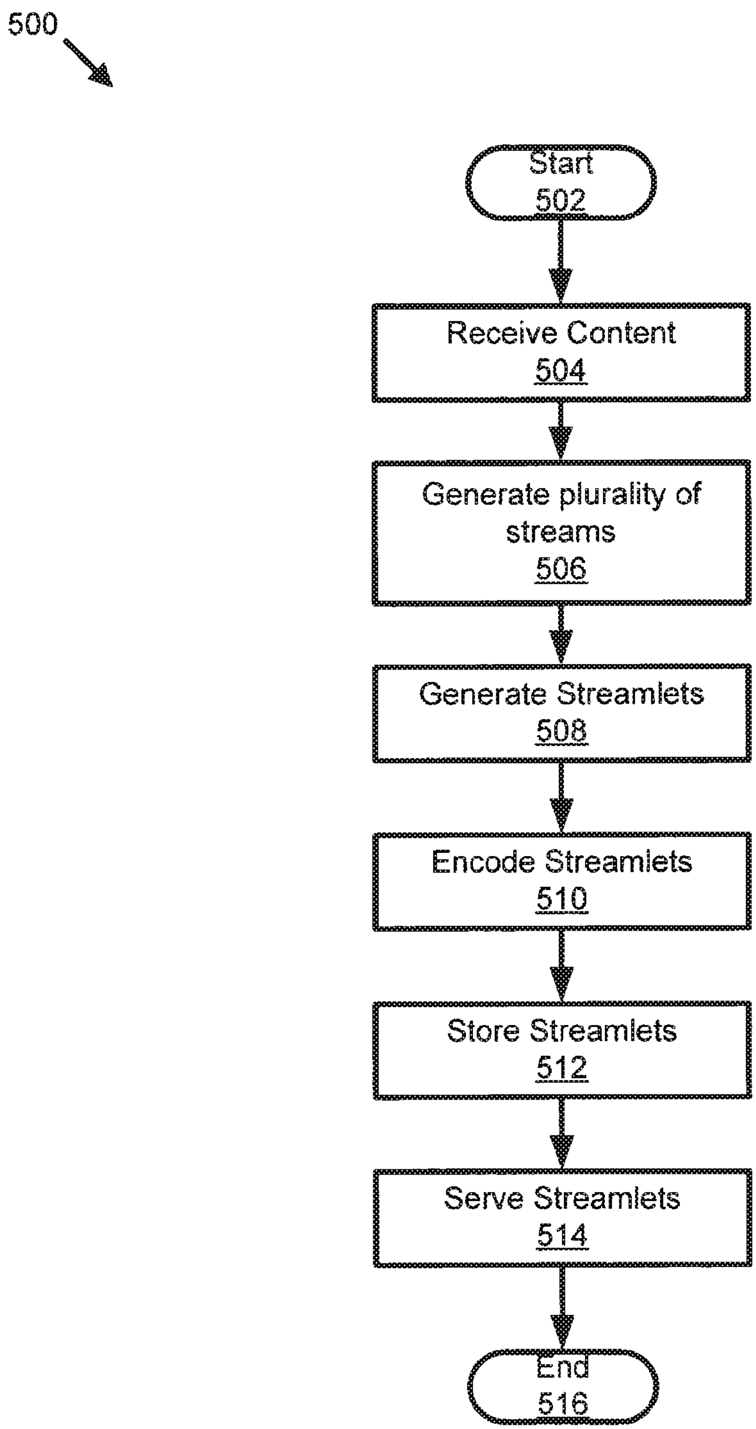
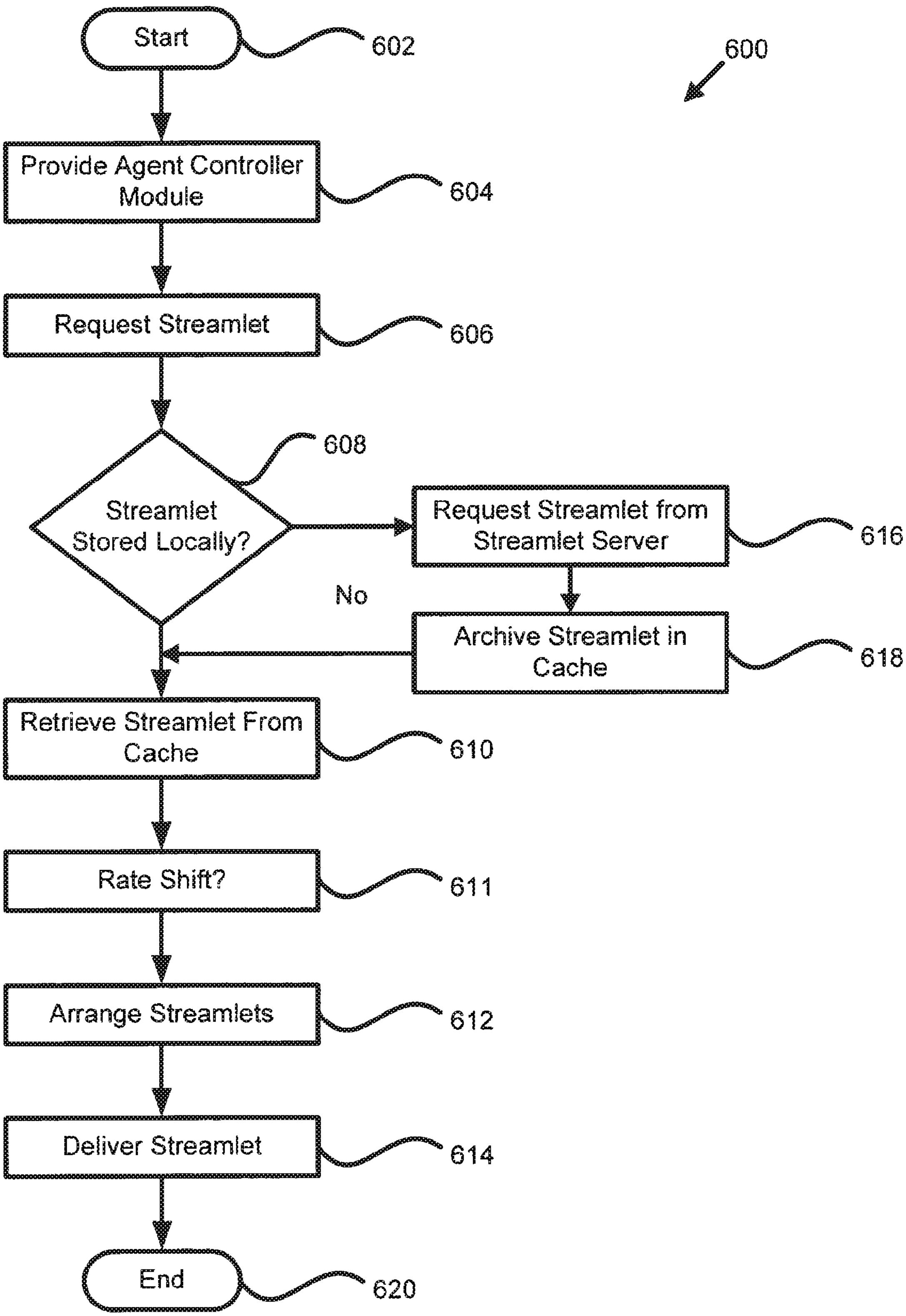

500
Start
502
Receive Content
504
Generate plurality of streams
506
Generate Streamlets
508
Encode Streamlets
510
Store Streamlets
512
Serve Streamlets
514
End
516

FIG. 5



Start
602
600
Provide Agent Controller Module
604
Request Streamlet
606
608
Streamlet Stored Locally?
Request Streamlet from Streamlet Server
616
No
Archive Streamlet in Cache
618
Retrieve Streamlet From Cache
610
Rate Shift?
611
Arrange Streamlets
612
Deliver Streamlet
614
End
620

FIG. 6

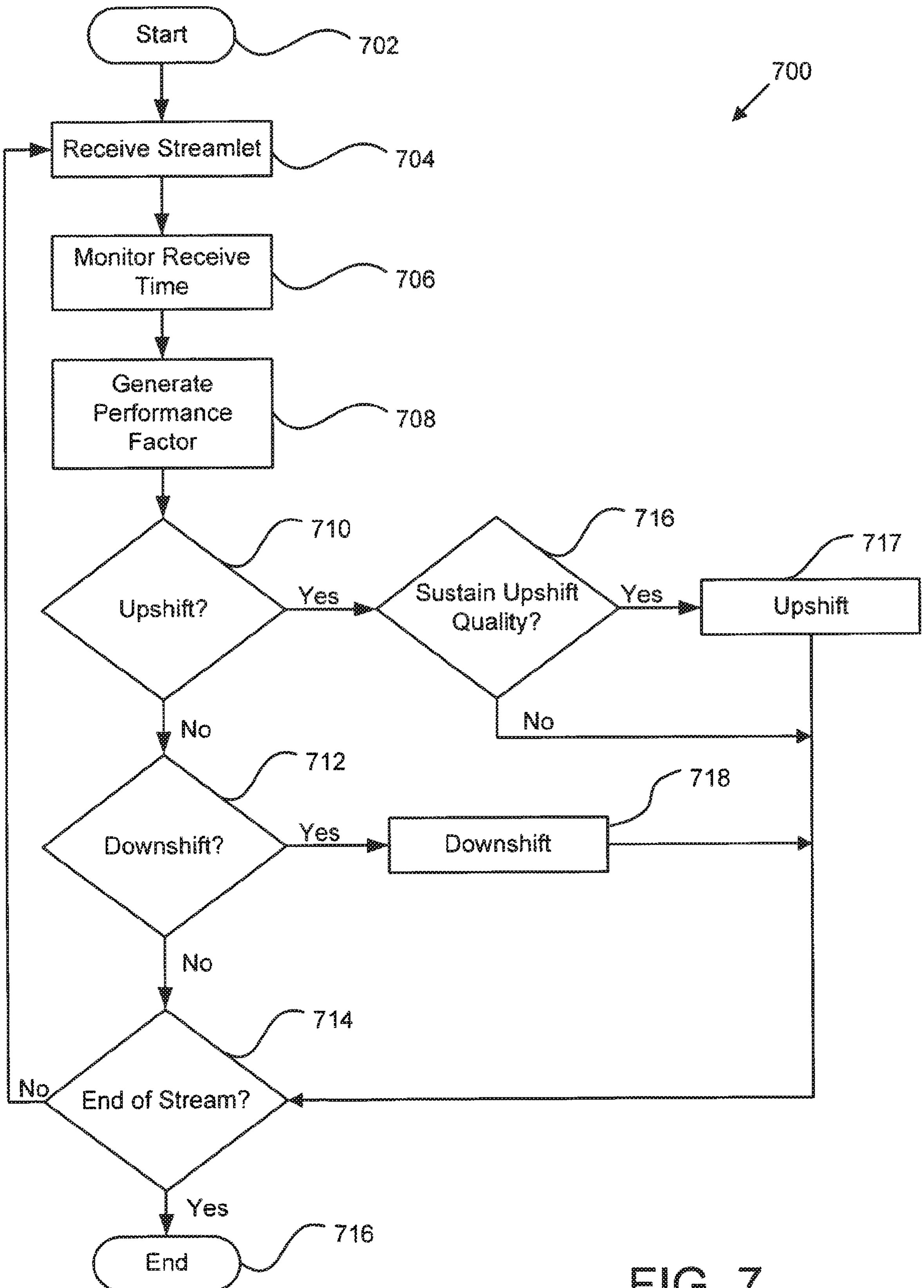

Start
702
700
Receive Streamlet
704
Monitor Receive Time
706
Generate Performance Factor
708
710
Upshift?
Yes
716
Sustain Upshift Quality?
Yes
717
Upshift
No
No
712
718
Downshift?
Yes
Downshift
No
714
No
End of Stream?
Yes
716
End

FIG. 7

APPARATUS, SYSTEM, AND METHOD FOR ADAPTIVE-RATE SHIFTING OF STREAMING CONTENT

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 15/207,172 (now U.S. Pat. No. 10,225,304), which is a continuation of U.S. patent application Ser. No. 14/516,303 (now U.S. Pat. No. 9,407,564), which is a continuation of U.S. patent application Ser. No. 11/116,783 (now U.S. Pat. No. 8,868,772), which claims benefit of United States Provisional Patent Application Ser. No. 60/566,831 entitled "APPARATUS, SYSTEM, AND METHOD FOR DYNAMIC RATE SHIFTING OF STREAMING CONTENT" and filed on Apr. 30, 2004 for R. Drew Major and Mark B. Hurst, which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to video streaming over packet switched networks such as the Internet, and more particularly relates to adaptive-rate shining of streaming content over such networks.

Description of the Related Art

The Internet is fast becoming a preferred method for distributing media files to end users. It is currently possible to download music or video to computers, cell phones, or practically any network capable device. Many portable media players are equipped with network connections and enabled lo play music or videos. The music or video files (hereinafter "media files") can be stored locally on the media player or computer, or streamed or downloaded from a server.

"Streaming media" refers to technology that delivers content at a rate sufficient for presenting the media to a user in real time as the data is received. The data may be stored in memory temporarily until played and then subsequently deleted. The user has the immediate satisfaction of viewing the requested content without waiting for the media file to completely download. Unfortunately, the audio/video quality that can be received for real time presentation is constrained by the available bandwidth of the user's network connection. Streaming may be used to deliver content on demand (previously recorded) or from live broadcasts.

Alternatively, media files may be downloaded and stored on persistent storage devices, such as hard drives or optical storage, for later presentation. Downloading complete media tiles can take large amounts of time depending on the network connection. Once downloaded, however, the content can be viewed repeatedly anytime or anywhere. Media files prepared for downloading usually are encoded with a higher quality audio/video than can be delivered in real time. Users generally dislike this option, as they tend to want to see or hear the media file instantaneously.

Streaming offers the advantage of immediate access to the content but currently sacrifices quality compared with downloading a file of the same content. Streaming also provides the opportunity for a user to select different content for viewing on an ad hoc basis, while downloading is by definition restricted to receiving a specific content selection in its entirety or not at all. Downloading also supports rewind, fast forward, and direct seek operations, while streaming is unable to fully support these functions. Streaming is also vulnerable to network failures or congestion.

Another technology, known as "progressive downloads," attempts to combine the strengths of the above two technologies. When a progressive download is initiated, the media file download begins, and the media player waits to begin playback until there is enough of the file downloaded that playback can begin with the hope that the remainder of the file will be completely downloaded before playback "catches up." This waiting period before playback can be substantial depending on network conditions, and therefore is not a complete or fully acceptable solution to the problem of media presentation over a network.

Generally, three basic challenges exist with regard to data transport streaming over a network such as the Internet that has a varying amount of data loss. The first challenge is reliability. Most streaming solutions use a TCP connection, or "virtual circuit," for transmitting data. A TCP connection provides a guaranteed delivery mechanism so that data sent from one endpoint will be delivered to the destination, even if portions are lost and retransmitted. A break in the continuity of a TCP connection can have serious consequences when the data must be delivered in real-time. When a network adapter detects delays or losses in a TCP connection, the adapter "backs off" from transmission attempts for a moment and then slowly resumes the original transmission pace. This behavior is an attempt to alleviate the perceived congestion. Such a slowdown is detrimental to the viewing or listening experience of the user and therefore is not acceptable.

The second challenge to data transport is efficiency. Efficiency refers to how well the user's available bandwidth is used for delivery of the content stream. This measure is directly related to the reliability of the TCP connection. When the TCP connection is suffering reliability problems, a loss of bandwidth utilization results. The measure of efficiency sometimes varies suddenly, and can greatly impact the viewing experience.

The third challenge is latency. Latency is the time measure form the client's point-of-view, of the interval between when a request is issued and the response data begins to arrive. This value is affected by the network connection's reliability and efficiency, and the processing time required by the origin to prepare the response. A busy or overloaded server, for example, will take more time to process a request. As well as affecting the start time of a particular request, latency has a significant impact on the network throughput of TCP.

From the foregoing discussion, it should be apparent that a need exists for an apparatus, system, and method that alleviate the problems of reliability, efficiency, and latency. Additionally, such an apparatus, system, and method would offer instantaneous viewing along with the ability to fast forward, rewind, direct seek, and browse multiple streams. Beneficially, such an apparatus, system, and method would utilize multiple connections between a source and destination, requesting varying bitrate streams depending upon network conditions.

SUMMARY OF THE INVENTION

The present invention has been developed in response to the present state of the art, and in particular, in response to the problems and needs in the art that have not yet been fully solved by currently available content streaming systems.

Accordingly, the present invention has been developed to provide an apparatus, system, and method for adaptive-rate content streaming that overcome many or all of the above-discussed shortcomings in the art.

The apparatus for adaptive-rate content streaming is provided with a logic unit containing a plurality of modules configured to functionally execute the necessary steps. These modules in the described embodiments include an agent controller module configured to simultaneously request a plurality of streamlets, the agent controller module further configured to continuously monitor streamlet requests and subsequent responses, and accordingly request higher or lower quality streamlets, and a staging module configured to stage the streamlets and arrange the streamlets for playback on a content player.

The apparatus is further configured, in one embodiment, to establish multiple Transmission Control Protocol (TCP) connections with a content server, and request streamlets of varying bitrates. Each streamlet may further comprise a portion of a content file. Additionally, the agent controller module may be configured to generate a performance factor according to responses from streamlet requests.

In a further embodiment, the agent controller module is configured to upshift to a higher quality streamlet when the performance factor is greater than a threshold, and the agent controller module determines the higher quality playback can be sustained according to a combination of factors. The factors may include an amount of contiguously available streamlets stored in the staging module, a minimum safety margin, and a current read ahead margin.

The agent controller module may be configured to downshift to a lower quality streamlet when the performance factor is less than a second threshold. Also, the agent controller module is further configured to anticipate streamlet requests and pre-request streamlets to enable fast-forward, skip randomly, and rewind functionality. In one embodiment, the agent controller module is configured to initially request low quality streamlets to enable instant playback of the content file, and subsequent upshifting according to the performance factor.

A system of the present invention is also presented to adaptive-rate content streaming In particular, the system, in one embodiment, includes a data communications network, and a content server coupled to the data communications network and having a content module configured to process content and generate a plurality of high and low quality streams. In one embodiment, each of the high and low quality streams may include a plurality of streamlets.

In a further embodiment, the system also includes an agent controller module configured to simultaneously request a plurality of streamlets, the agent controller module further configured to continuously monitor streamlet requests and subsequent responses, and accordingly request higher or lower quality streamlets, and a staging module configured to stage the streamlets and arrange the streamlets tor playback on a content player.

A method of the present invention is alto presented for adaptive-rate content streaming. The method in the disclosed embodiments substantially includes the steps necessary to carry out the functions presented above with respect to the operation of the described apparatus and system. In one embodiment, the method includes simultaneously requesting a plurality of streamlets, continuously monitoring streamlet requests and subsequent responses, and accordingly requesting higher or lower quality streamlets, and staging the streamlets and arranging the streamlets for playback on a content player.

In a further embodiment, the method may include establishing multiple Transmission Control Protocol (TCP) connections with a content server, and requesting streamlets of varying bitrates. Also, the method may include generating a performance factor according to responses from streamlet requests, upshifting to a higher quality streamlet when the performance factor is greater than a threshold, and determining if the higher quality playback can be sustained. Furthermore, the method may include downshifting to a lower quality streamlet when the performance factor is less than a second threshold.

In one embodiment, the method includes anticipating streamlet requests and pre-requesting streamlets to enable fast-forward, skip randomly, and rewind functionality. The method may also comprise initially requesting low quality streamlets to enable instant playback of a content file, and subsequent upshifting according to the performance factor.

Reference throughout this specification to features, advantages, or similar language does not imply that all of the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussion of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

Furthermore, the described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize that the invention may be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

These features and advantages of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the advantages of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings, in which:

FIG. **1** is a schematic block diagram illustrating one embodiment of a system for adaptive rate shifting of streaming content in accordance with the present invention;

FIG. **2***a* is a schematic block diagram graphically illustrating one embodiment of a content file in accordance with the present invention;

FIG. **2***b* is a schematic block diagram illustrating one embodiment of a plurality of streams having varying degrees of quality and bandwidth in accordance with the present invention;

FIG. **2c** is a schematic block diagram illustrating one embodiment of a stream divided into a plurality of streamlets in accordance with the present invention;

FIG. **3** is a schematic block diagram illustrating one embodiment of a content module in accordance with the present invention;

FIG. **4** is a schematic block diagram graphically illustrating one embodiment of a client module in accordance with the present invention.

FIG. **5** is a schematic flow chart diagram illustrating one embodiment of a method for processing content in accordance with the present invention;

FIG. **6** is a schematic flow chart diagram illustrating one embodiment of a method for playback of a plurality of streamlets in accordance with the present invention; and

FIG. **7** is a schematic flow chart diagram illustrating one embodiment of a method for requesting streamlets within an adaptive-rate content streaming environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Many of the functional units described in this specification have been labeled as modules, in order to more particularly emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom VLSI circuits or gate arrays, off-the-shelf semi conductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices or the like.

Modules may also be implemented in software for execution by various types of processors. An identified module of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions which may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may comprise disparate instructions stored in different locations which, when joined logically together, comprise the module and achieve the stated purpose for the module.

Indeed, a module of executable code may be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

Reference throughout this specification to “one embodiment,” “an embodiment,” or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases “in one embodiment,” “in an embodiment,” and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

Reference to a signal bearing medium may take any form capable of generating a signal, causing a signal to be generated, or causing execution of a program of machine-readable instructions on a digital processing apparatus. A signal bearing medium may be embodied by a transmission line, a compact disk, digital-video disk, a magnetic tape, a Bernoulli drive, a magnetic disk, a punch card, flash memory, integrated circuits, or other digital processing apparatus memory device.

Furthermore, the described features, structures, or characteristics of the invention may be combined in any suitable manner in one or more embodiments. In the following description, numerous specific details are provided, such as examples of programming, software modules, user selections, network transactions, database queries, database structures, hardware modules, hardware circuits, hardware chips, etc., to provide a thorough understanding of embodiments of the invention. One skilled in the relevant art will recognize, however, that the invention may be practiced without one or more of the specific details, or with other methods, components, materials, and so forth. In other instances, well-known structures, materials, or operations are not shown or described in detail to avoid obscuring aspects of the invention.

FIG. **1** is a schematic block diagram illustrating one embodiment of a system **100** for dynamic rate shifting of streaming content in accordance with the present invention. In one embodiment, the system **100** comprises a content server **102** and an end user **104**. The content server **102** and the end user station **104** may be coupled by a data communications network. The data communications network may include the Internet **106** and connections **108** to the Internet **106**. Alternatively, the content server **102** and the end user **104** may be located on a common local area network, wireless area network, cellular network, virtual local area network, or the like. The end user station **104** may comprise a personal computer (PC), an entertainment system configured to communicate over a network, or a portable electronic device configured lo present content.

In the depicted embodiment, the system **100** also includes a publisher **110**, and a web server **116**. The publisher **110** may be a creator or distributor of content. For example, if the content lobe streamed were a broadcast of a television program, the publisher **110** may be a television or cable network channel such as NBC®, or MTV®. Content may be transferred over the Internet **106** to the content server **102**, where the content is received by a content module **112**. The content module **112** may be configured to receive, process, and store content. In one embodiment, processed content is accessed by a client module **114** configured to play the content on the end user station **104**. In a further embodiment, the client module **114** is configured to receive different portions of a content stream from a plurality of legations simultaneously. For example, the client module **114** may request and receive content from any of the plurality of web servers **116**.

FIG. **2a** is a schematic block diagram graphically illustrating one embodiment of a content file **200**. In one embodiment, the content file **200** is distributed by the publisher **110**. The content file **200** may comprise a television broadcast, sports event, movie, music, concert, etc. The content file **200** may also be live or archived content. The content file **200** may comprise uncompressed video and audio, or alternatively, video or audio. Additionally, the content file **200** may be compressed. Examples of a compressed content file **200** include, but are not limited to, DivX®, Windows Media Video 9®, Quicktime 6.5 Sorenson 3®, or Quicktime 6.5/MPEG-4® encoded content.

FIG. **2b** is a schematic block diagram illustrating one embodiment of a plurality of streams **202** having varying degrees of quality and bandwidth. In one embodiment, the plurality of streams **202** comprises a low quality stream **204**,

a medium quality stream **206**, and a high quality stream **208**. Each of the streams **204**, **206**, **208** is a copy of the content file **200** encoded and compressed to varying bit rates. For example, the low quality stream **204** may be encoded and compressed to a bit rate of 100 kilobits per second (kbps), the medium quality stream **206** may be encoded and compressed to a bit rate of 200 kbps, and the high quality stream **208** may be encoded and compressed to 600 kbps.

FIG. **2***c* is a schematic block diagram illustrating one embodiment of a stream **210** divided into a plurality of streamlets **212**. As used herein, streamlet refers to any sized portion of the content file **200**. Each streamlet **212** may comprise a portion of the content contained in stream **110**, encapsulated as an independent media object. The content in a streamlet **212** may have a unique time index in relation to the beginning of the content contained in stream **210**. In one embodiment, the content contained in each streamlet **212** has a duration of two seconds. For example, streamlet 0 may have a time index of 00:00 representing the beginning of content playback, and streamlet 1 may have a time index of 00:02, and so on. Alternatively, the time duration of the streamlets **212** may be any duration smaller than the entire playback duration of the content in stream **210**. In a further embodiment, the streamlets **212** may be divided according to file size instead of a time index.

FIG. **3** is a schematic block diagram illustrating in greater detail one embodiment of the content module **112** in accordance with the present invention. The content module **112** may comprise a stream module **302**, a streamlet module **304**, an encoder module **306**, a streamlet database **308**, and the web server **116**. In one embodiment, the stream module **302** is configured to receive the content file **200** from the publisher **110** and generate the plurality of streams **202** of varying qualities. The original content file **200** from the publisher may be digital in form and may comprise content having a high bit rate such as, for example, 2 mbps. The content may be transferred from the publisher **110** to the content module **112** over the Internet **106**. Such transfers of data are well known in the art and do not require further discussion herein. Alternatively, the content may comprise a captured broadcast.

In the depicted embodiment, the plurality of streams **202** may comprise the low quality stream **204**, the medium quality stream **206**, and the high quality stream **208**. Alternatively, the plurality of streams **202** may comprise any number of streams deemed necessary to accommodate end user bandwidth. The streamlet module **304** may be configured to receive the plurality of streams **202** from the stream module and generate a plurality of streams **312**, each stream comprising a plurality of streamlets **212**. As described with reference to FIG. **2***c,* each streamlet **212** may comprise a pre-defined portion of the stream. The encoder module **306** is configured to encode each streamlet from the plurality of streams **312** and store the streamlets in the streamlet database **308** The encoding module **306** may utilize encoding schemes such as DivX®, Windows Media Video 9®, Quicktime 6.5 Sorenson 3®, or Quicktime 6.5/MPEG-4®. Alternatively, a custom encoding scheme may be employed.

The content module **112** may also include a metadata module **312** and a metadata database **314**. In one embodiment, metadata comprises static searchable content information. For example, metadata includes, but is not limited to, air date of the content, title, actresses, actors, length, and episode name. Metadata is generated by the publisher **110**, and may be configured to define an end user environment. In one embodiment, the publisher **100** may define an end user navigational environment for the content including menus, thumbnails, sidebars, advertising, etc. Additionally, the publisher **110** may define functions such as fast forward, rewind, pause, sad play that may be used with the content file **200**. The metadata module **312** is configured to receive the metadata from the publisher **110** and store the metadata in the metadata database **314**. In a further embodiment, the metadata module **312** is configured to interface with the client module **114**, allowing the client module **114** to search for content based upon at least one of a plurality of metadata criteria. Additionally, metadata may be generated by the content module **112** through automated processes or manual definition.

Once the streamlets **212** have been received and processed, the client module **114** may request streamlets **212** using HTTP from the web server **116**. Such use of client side initiated requests requires no additional configuration of firewalls. Additionally, since the client module **114** initiates the request, the web server **116** is only required to retrieve and serve the requested streamlet. In a further embodiment, the client module **114** may be configured to retrieve streamlets **212** from a plurality of web servers **310**. Each web server **116** may be located in various locations across the Internet **106**. The streamlets **212** are essentially static files. As such, no specialized media server or server-side intelligence is required for a client module **114** to retrieve streamlets **212**. Streamlets **212** may be served by the web server **116** or cached by cache servers of Internet Service Providers (ISPs), or any ether network infrastructure operators, and served by the cache server. Use of cache servers is well known to those skilled in the art, and will not be discussed further herein. Thus, a highly scalable solution is provided that is not hindered by massive amounts of client module **114** requests to the web server **116** at any specific location.

FIG. **4** is a schematic block diagram graphically illustrating one embodiment of a client module **114** in accordance with the present invention. The client module **114** may comprise an agent controller module **402**, a streamlet cache module **404**, and a network controller module **406**. In one embodiment, the agent controller module **402** is configured to interface with a viewer **408**, and transmit streamlets **212** to the viewer **408**. In a further embodiment, the client module **114** may comprise a plurality of agent controller modules **402**. Each agent controller module **402** may be configured to interface with one viewer **408**. Alternatively, the agent controller module **402** may be configured to interface with a plurality of viewers **408**. The viewer **408** may be a media player (not shown) operating on a PC or handheld electronic device.

The agent controller module **402** is configured to select a quality level of streamlets to transmit to the viewer **408** The agent controller module **402** requests lower or higher quality streams based upon continuous observation of time intervals between successive receive times of each requested streamlet. The method of requesting higher or lower quality streams will be discussed in greater detail below with reference to FIG. **7**.

The agent controller module **402** may be configured to receive user commands from the viewer **408**. Such commands may include play, fast forward, rewind, pause, and stop. In one embodiment, the agent controller module **402** requests streamlets **212** from the streamlet cache module **404** and arranges the received streamlets **212** in a staging module **409**. The staging module **409** may be configured to arrange the streamlets **212** in order of ascending playback time. In the depicted embodiment, the streamlets **212** are numbered 0, 1, 2, 3, 4, etc. However, each streamlet **212** may be identified with a unique filename.

Additionally, the agent controller module **402** may be configured to anticipate streamlet **212** requests and pre-request streamlets **212**. By pre-requesting streamlets **212**, the user may fast-forward, skip randomly, or rewind through the content and experience no buffering delay. In a further embodiment, the agent controller module **402** may request the streamlets **212** that correspond to time index intervals of 30 seconds within the total play time of the content. Alternatively, the agent controller module **402** may request streamlets at any interval less than the length of the time index. This enables a "fast-start" capability with no buffering wait when starting or fast-forwarding through content file **200**. In a further embodiment, the agent controller module **402** may be configured to pre-request streamlets **212** corresponding to specified index points within the content or within other content in anticipation of the end user **104** selecting new content to view.

In one embodiment, the streamlet cache module **404** is configured to receive streamlet **212** requests from the agent controller module **402**. Upon receiving a request, the streamlet cache module **404** first checks a streamlet cache **410** to verify if the streamlet **212** is present. In a further embodiment, the streamlet cache module **404** handles streamlet **212** requests from a plurality of agent controller modules **402**. Alternatively, a streamlet cache module **404** may be provided for each agent controller module **402**. If the requested streamlet **212** is not present in the streamlet cache **410** the request is passed to the network controller module **406**. In order to enable fast forward and rewind capabilities, the streamlet cache module **404** is configured to store the plurality of streamlets **212** in the streamlet cache **410** for a specified time period after the streamlet **212** has been viewed. However, once the streamlets **212** have been deleted, they maybe requested again from the web server **116**.

The network controller module **406** may be configured to receive streamlet requests from the streamlet cache module **404** and open a connection to the web server **116** or other remote streamlet **212** database (not shown). In one embodiment, the network controller module **406** opens a TCP/IP connection to the web server **116** and generates a standard HTTP GET request for the requested streamlet **212**. Upon receiving the requested streamlet **212**, the network controller module **406** passes the streamlet **212** to the streamlet cache module **404** where it is stored in the streamlet cache **410**. In a further embodiment, the network controller module **406** is configured to process and request a plurality of streamlets **212** simultaneously. The network controller module **406** may also be configured to request a plurality of streamlets, where each streamlet **212** is subsequently requested in multiple parts.

In a further embodiment, streamlet requests may comprise requesting pieces of any streamlet file. Splitting the streamlet **212** into smaller pieces or portions beneficially allows for an increased efficiency potential, and also eliminates problems associated with multiple full-streamlet requests sharing the bandwidth at any given moment. This is achieved by using parallel TCP/IP connections for pieces of the streamlets **212**. Consequently, efficiency and network loss problems are overcome, and the streamlets arrive with more useful and predictable timing.

In one embodiment, the client module **114** is configured to use multiple TCP connections between the client module **114** and the web server **116** or web cache. The intervention of a cache may be transparent to the client or configured by the client as a forward cache. By requesting more than one streamlet **212** at a time in a manner referred to as "parallel retrieval," or more than one pan of a streamlet **212** at a time, efficiency is raised significantly and latency is virtually eliminated. In a further embodiment, the client module allows a maximum of three outstanding streamlet **212** requests. The client module **114** may maintain additional open TCP connections as spares to be available should another connection fail. Streamlet **212** requests are rotated among all open connections to keep the TCP flow logic for any particular connection from failing into a slow-start or close mode, if the network controller module **406** has requested a streamlet **212** in multiple parts, with each part requested on mutually independent TCP/IP connections, the network controller module **406** reassembles the parts to present a complete streamlet **212** for use by all other components of the client module **114**.

When a TCP connection fails completely, a new request may be sent on a different connection for the same streamlet **212**. In a further embodiment, if a request is not being satisfied in a timely manner, a redundant request may be sent on a different connection for the same streamlet **212**. If the first streamlet request's response arrives before the redundant request response, the redundant request can be aborted. If the redundant request response arrives before the first request response, the first request may be aborted.

Several streamlet **212** requests may be sent on a single TCP connection, and the responses are caused to flow back in matching order along the same connection. This eliminates all but the first request latency. Because multiple responses are always being transmitted, the processing latency of each new streamlet **212** response after the first is not a factor in performance. This technique is known in the industry as "pipelining." Pipelining offers efficiency in request-response processing by eliminating most of the effects of request latency. However, pipelining has serious vulnerabilities. Transmission delays affect all of the responses. If the single TCP connection fails, all of the outstanding requests and responses are lost. Pipelining causes a serial dependency between the requests.

Multiple TCP connections may be opened between the client module **114** and the web server **116** to achieve the latency-reduction efficiency benefits of pipelining while maintaining the independence of each streamlet **212** request. Several streamlet **212** requests may be sent concurrently, with each request being sent on a mutually distinct TCP connection This technique is labeled "virtual pipelining" and is an innovation of the present invention. Multiple responses may be in transit concurrently, assuring that communication bandwidth between the client module **114** and the web server **116** is always being utilized. Virtual pipelining eliminates the vulnerabilities of traditional pipelining. A delay in or complete failure of one response does not affect the transmission of other responses because each response occupies an independent TCP connection. Any transmission bandwidth not in use by one of multiple responses (whether due to delays or TCP connection failure) may be utilized by other outstanding responses.

A single streamlet **212** request may be issued for an entire streamlet **212**, or multiple requests may be issued, each for a different part or portion of the streamlet. If the streamlet is requested in several parts, the parts may be recombined by the client module **114** streamlet.

In order to maintain a proper balance between maximized bandwidth utilization and response time, the issuance of new streamlet requests must be timed such that the web server **116** does not transmit the response before the client module **114** has fully received a response to one of the previously outstanding streamlet requests. For example, if three stream-

let **212** requests are outstanding, the client module **114** should issue the next request slightly before one of the three responses is fully received and "out of the pipe." In other words, request timing is adjusted to keep three responses in transit. Sharing of bandwidth among four responses diminishes the net response time of the other three responses. The timing adjustment may be calculated dynamically by observation, and the request timing adjusted accordingly to maintain the proper balance of efficiency and response times.

The schematic flow chart diagrams that follow are generally set forth as logical flow chart diagrams. As such, the depicted order and labeled steps are indicative of one embodiment of the presented method. Other steps and methods may be conceived that are equivalent in function, logic, or effect to one or more steps, or portions thereof, of the illustrated method. Additionally, the format and symbols employed are provided to explain the logical steps of the method and are understood not to limit the scope of the method. Although various arrow types and line types may be employed in the flow chart diagrams, they are understood not to limit the scope of the corresponding method. Indeed, some arrows or other connectors may be used to indicate only the logical flow of the method. For instance, an arrow may indicate a waiting or monitoring period of unspecified duration between enumerated steps of the depicted method. Additionally, the order in which a particular method occurs may or may not strictly adhere to the order of the corresponding steps shown.

FIG. **5** is a schematic (low chart diagram illustrating one embodiment of a method **500** for processing content in accordance with the present invention. In one embodiment the method **500** starts **502**, and the content module **112** receives **504** content from the publisher **110**. Receiving content **504** may comprise receiving **504** a digital copy of the content file **200**, or digitizing a physical copy of the content file **200**. Alternatively, receiving **504** content may comprise capturing a radio or television broadcast. Once received **504**, the stream module **302** generates **506** a plurality of streams **202**, each stream **202** having a different quality. The quality may be predefined, or automatically set according to end user bandwidth, or in response to pre-designated publisher guidelines.

The streamlet module **304** receives the streams **202** and generates **508** a plurality of streamlets **212**. In one embodiment, generating **508** streamlets comprises dividing the stream **202** into a plurality of two second streamlets **212**. Alternatively, the streamlets may have any length less than or equal to the length of the stream **202** The encoder module **306** then encodes **510** the streamlets according to a compression algorithm. In a further embodiment, the algorithm comprises a proprietary codec such as WMV9®. The encoder module **306** then stores **512** the encoded streamlets in the streamlet database **308**. Once stored **512**, the web server **116** may then serve **514** the streamlets. In one embodiment, serving **514** the streamlets comprises receiving streamlet requests from the client module **114**, retrieving the requested streamlet from the streamlet database **308**, and subsequently transmitting the streamlet to the client module **114**. The method **500** then ends **516**.

FIG. **6** is a schematic flow chart diagram illustrating one embodiment of a method **600** for viewing a plurality of streamlets in accordance with the present invention. The method **600** starts and an agent control module **402** is provided **604** and associated with a viewer **408** and provided with a staging module **409**. The agent controller module **402** then requests **606** a streamlet from the streamlet cache module **404**. Alternatively, the agent controller module **402** may simultaneously request **606** a plurality of streamlets from the streamlet cache module **404**. If the streamlet is stored **608** locally in the streamlet cache **410**, the streamlet cache module **404** retrieves **610** the streamlet and sends the streamlet to the agent controller module **402**. Upon retrieving **610** or receiving a streamlet, the agent controller module **402** makes **611** a determination of whether or not to shift to a higher or lower quality stream **202**. This determination will be described below in greater detail with reference to FIG. **7**.

In one embodiment, the staging module **409** then arranges **612** the streamlets into the proper order, and the agent controller module **402** delivers **614** the streamlets to the viewer **408**. In a further embodiment, delivering **614** streamlets to the end user comprises playing video and or audio streamlets on the viewer **408**. If the streamlets are not stored **608** locally, the streamlet request is passed to the network controller module **406**. The network controller module **406** then requests **616** the streamlet from the web server **116**. Once the streamlet is received, the network controller module **406** passes the streamlet to the streamlet cache module **404** The streamlet cache module **404** archives **618** the streamlet. Alternatively, the streamlet cache module **404** then archives **618** the streamlet and passes the streamlet to the agent controller module **402**, and the method **600** then continues from operation **610** as described above.

Referring now to FIG. **7**, shown therein is a schematic flow chart diagram illustrating one embodiment of a method **700** for requesting streamlets within a adaptive-rate shifting content streaming environment in accordance with the present invention. The method **700** may be used in one embodiment as the operation **611** of FIG. **6**. The method **700** starts and the agent controller module **402** receives **704** a streamlet as described above with reference to FIG. **6**. The agent controller module **402** then monitors **706** the receive time of the requested streamlet. In one embodiment, the agent controller module **402** monitors the time intervals Δ between successive receive times for each streamlet response. Ordering of the responses in relation to the order of their corresponding requests is not relevant.

Because network behavioral characteristics fluctuate, sometimes quite suddenly, any given Δ may vary substantially from another. In order to compensate for this fluctuation, the agent controller module **402** calculates **708** a performance ratio r across a window of n samples for streamlets of playback length S. In one embodiment, the performance ratio r is calculated using the equation

$$r = S\frac{n}{\sum_{i=1}^{n} \Delta_i}.$$

Due to multiple simultaneous streamlet processing, and in order to better judge the central tendency of the performance ration r, the agent control module **402** may calculate a geometric mean, or alternatively an equivalent averaging algorithm, across a window of size m, and obtain a performance factor φ:

$$\varphi_{current} = \left(\prod_{j=1}^{m} r_j\right)^{\frac{1}{m}}.$$

The policy determination about whether or not to upshift **710** playback quality begins by comparing $\varphi_{current}$ with a trigger threshold Θ_{up}. If $\varphi_{current} \geq \Theta_{up}$, then an up shift to the next higher quality stream may be considered **716**. In one embodiment, the trigger threshold Θ_{up} is determined by a combination of factors relating to the current read ahead margin (i.e. the amount of contiguously available streamlets that have been sequentially arranged by the staging module **409** for presentation at the current playback time index), and a minimum safety margin. In one embodiment, the minimum safety margin may be 24 seconds. The smaller the read ahead margin, the larger Θ_{up} is to discourage upshifting until a larger read ahead margin may be established to withstand network disruptions. If the agent controller module **402** is able to sustain **716** upshift quality, then the agent controller module **402** will upshift **717** the quality and subsequently request higher quality streams. The determination of whether use of the higher quality stream is sustainable **716** is made by comparing an estimate of the higher quality stream's performance factor, φ_{higher}, with Θ_{up}. If $\varphi_{higher} \geq \Theta_{up}$ then use of the higher quality stream is considered sustainable. If the decision of whether or not the higher stream rate is sustainable **716** is "no," the agent control module **402** will not attempt to upshift **717** stream quality. If the end of the stream has been reached **714**, the method **618** ends **716**.

If the decision on whether or not to attempt upshift **710** is "no", a decision about whether or not to downshift **712** is made. In one embodiment, a trigger threshold Θ_{down} is defined in a manner analogous to Θ_{up}. If $\varphi_{current} > \Theta_{down}$ then the stream quality may be adequate, and the agent controller module **402** does not downshift **718** stream quality. However, if $\varphi_{current} \leq \Theta_{down}$, the agent controller module **402** does downshift **718** the stream quality. If the end of the stream has not been reached **714**, the agent controller module **402** begins to request and receive **704** lower quality streamlets and the method **618** starts again. Of course, the above described equations and algorithms are illustrative only, and may be replaced by alternative streamlet monitoring solutions.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. An apparatus for rendering a video that is adaptively received as a digital stream from a video server over a network, the apparatus comprising;

a media player operating on the apparatus, wherein the media player is configured to stream the video from the video server via at least one transmission control protocol (TCP) connection over the network, wherein the video server stores multiple different copies of the video encoded at different bit rates as multiple sets of streamlets, wherein each of the streamlets yields a different portion of the video on playback, wherein the streamlets across the different copies yield the same portions of the video on playback, and wherein the streamlets in the different copies are aligned in time such that the streamlets that play back the same portion of the video for the different copies each begin at the same playback time in relation to the beginning of the video, and wherein the media player streams the video by:

requesting sequential streamlets of one of the copies from the video server according to the playback times of the streamlets by transmitting hypertext transport protocol (HTTP) GET requests that identify the selected streamlets stored by the video server, wherein the sequential streamlets are selected by the media player from the based upon successive determinations to shift the playback quality to a higher or lower quality one of the different copies of the video;

repeatedly generating, by the media player, a factor relating to the performance of the network that is indicative of an ability to sustain the streaming of the video;

adapting the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the streamlets of the highest quality copy of the video that is determined to be sustainable at that time; and

presenting the video for playback by providing the requested streamlets in order of ascending start time.

2. The apparatus of claim **1**, wherein the apparatus is configured to establish multiple Transmission Control Protocol (TCP) connections with a content server, and request streamlets of varying bitrates.

3. The apparatus of claim **1**, wherein each streamlet further comprises a portion of a content file provided by the server.

4. The apparatus of claim **1**, wherein the requesting the sequential streamlets comprises the apparatus transmitting hypertext transport protocol (HTTP) GET requests for selected streamlets, wherein each of the HTTP GET requests identifies the separate file stored by the video server that corresponds to the requested streamlet.

5. The apparatus of claim **1** wherein each of the streamlets of each of the different copies is independently requestable and playable by the apparatus.

6. The apparatus of claim **4**, wherein the requesting of the sequential streamlets comprises the end user device transmitting hypertext transport protocol (HTTP) GET requests for selected streamlets.

7. The apparatus of claim **1** wherein each of the streamlets in each of the plurality of different copies is a separate file stored by the video server.

8. The apparatus of claim **1** wherein the media player upshifts to a higher quality one of the different copies when the factor is greater than a first threshold and downshifts to a lower quality one of the different copies when the factor is less than a second threshold.

9. The apparatus of claim **1**, wherein the requesting the sequential streamlets comprises the apparatus transmitting hypertext transport protocol (HTTP) GET requests for selected streamlets, and wherein each of the HTTP GET requests identifies a separately-identifiable portion of the one or more files that corresponds to the requested streamlet.

10. The apparatus of claim **1**, wherein the apparatus is configured to initially request low quality streamlets to enable instant playback of the content file, and subsequent upshifting according to the performance factor.

11. The end user device of claim **1** wherein each of the streamlets in each of the plurality of different copies is a separately-identifiable portion of one or more files stored by the video server.

12. The apparatus of claim **1** wherein the apparatus is a mobile computing device comprising a processor and a non-transitory data storage.

13. A method executable by an end user device to stream a video received via a connection with a server over a network, the method comprising:

requesting, by the end user device, wherein the end user device streams the video from the video server via at least one transmission control protocol (TCP) connection over the network, a plurality of sequential streamlets of one of the copies from the server based on playback times of the streamlets wherein multiple different copies of the video encoded at different bit rates are stored as multiple sets of streamlets on the server, wherein each of the streamlets yields a different portion of the video on playback, wherein the streamlets across the different copies yield the same portions of the video on playback, and wherein each of the streamlets comprises a playback time such that each of the streamlets for each of the different copies that encode the same portion of the video begins at the same playback time in relation to the beginning of the video, wherein the end user device requests the streamlets by transmitting hypertext transport protocol (HTTP) GET requests that each identify one of the requested streamlets stored by the server; and wherein the end user device streams the video by:

repeatedly generating, by the end user device, a factor that is indicative of an ability to sustain the streaming of the video;

making successive determinations by the end user device to shift the playback quality based on the factor to achieve continuous playback of the video using the streamlets of the highest quality copy determined sustainable at that time; and

presenting the video by playing back the requested media streamlets on the end user device in order of ascending playback time.

14. The method of claim **13** wherein the making of the successive determinations to shift comprises upshifting to a higher quality one of the different copies when the at least one factor is greater than a first threshold and downshifting to a lower quality one of the different copies when the at least one factor is less than a second threshold.

15. The method of claim **14** wherein each of the streamlets of each of the different copies is independently requestable and playable by the end user device.

16. The method of claim **13** wherein each of the streamlets in each of the plurality of different copies is a separately-identifiable portion of one or more files stored by the video server.

17. The method of claim **16** wherein the requesting of the sequential streamlets comprises the end user device transmitting hypertext transport protocol (HTTP) GET requests for selected streamlets, and wherein each of the HTTP GET requests identifies the separately-identifiable portion of the one or more files that corresponds to the requested streamlet.

18. The method of claim **13**, wherein the video captures a live event, and wherein the streamlets of the different copies are available to the end user device while the live event is occurring.

* * * * *

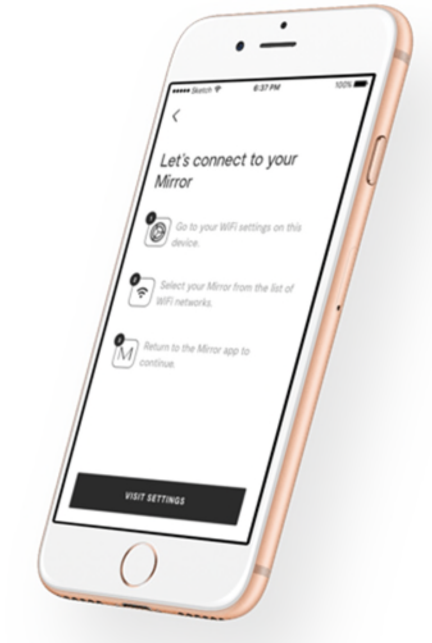
EXHIBIT D-1

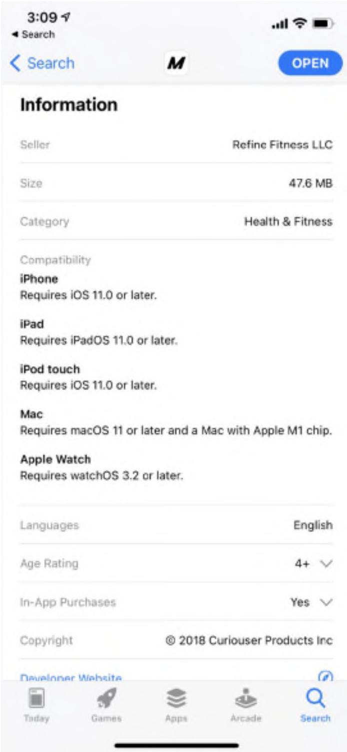
U.S. Patent No. 10,757,156 to Mirror

The following claim chart shows exemplary aspects of the Mirror Application and Mirror Device that infringe the claim below. The chart is exemplary and should not be read to limit DISH's claims against Mirror to the specific products or services described below. The chart should also not be read to limit DISH's claims to the patent claim charted below. Nor should the chart below be read to limit how the Mirror Application and Mirror Devices infringe the claim below.

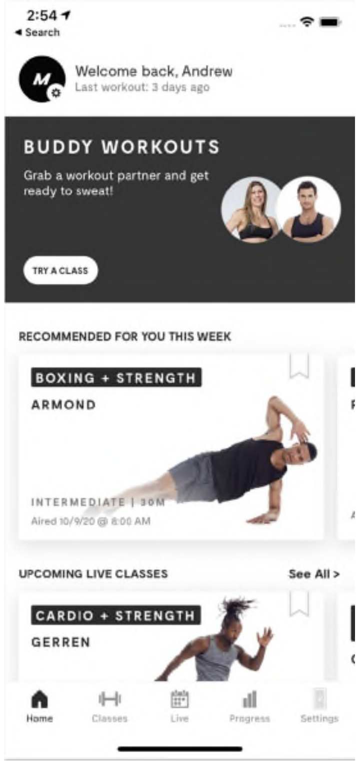
Claim	Claim Limitation	Infringement Analysis
1	An apparatus for rendering a video that is adaptively received as a digital stream from a video server over a network, the apparatus comprising;	The Mirror Application is software that causes "[a]n apparatus" to "render[] a video that is adaptively received as a digital stream from a video server of a network." The Mirror Application is executable by an "apparatus," and it renders or plays back video that is adaptively received as a digital stream from a "video server." The video and digital stream is obtained by the Mirror Application from a "video server," over a network. The images in this chart of the Mirror Application are from the Mirror Application running on an Apple iPhone XS (Mirror's iOS Application) and connected to the Internet via TCP/IP protocols. In addition, the Mirror Application is available to run on other devices. Unless otherwise noted, each of these devices is an "apparatus" according to this claim preamble. **MIRROR DIGITAL OVERVIEW** Access all of MIRROR's Live and On Demand Classes 24/7 from your phone, tablet, and smart TV devices. This feature is included for free with the MIRROR Membership and access is available immediately (whether you already have your Mirror or are awaiting delivery). Please note, it is not currently possible to purchase MIRROR Digital if you do not own a Mirror. https://mirror.kustomer.help/en_us/mirror-digital-overview-B1pnVd8UU

USP 10,757,156 to Mirror

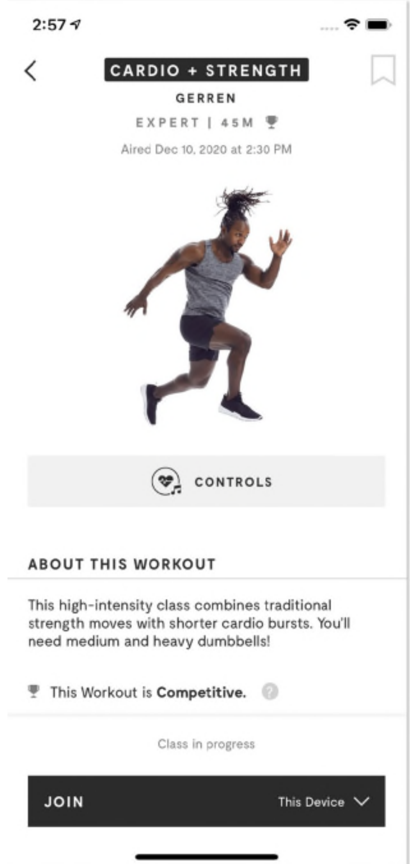
Claim	Claim Limitation	Infringement Analysis
		GET THE MIRROR APP To get started setting up your Mirror, you need to download the MIRROR app from the Apple App Store or Google Play Store. The app will take you through everything you need to know. Available on the App Store Get it on Google play Need help? Email us at hello@mirror.co  https://www.mirror.co/app. **MIRROR APP** The MIRROR App allows you to access and customize the Mirror experience. The MIRROR App is available for both iOS and Android! • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-app--S1dDC_tYm.

Claim	Claim Limitation	Infringement Analysis
		 Source: Apple App Store

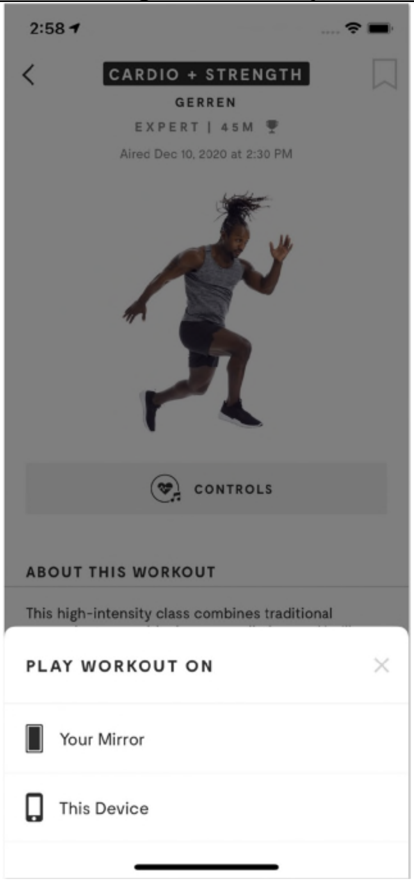
USP 10,757,156 to Mirror

Claim	Claim Limitation	Infringement Analysis
		When launched, the Mirror Application displays a main menu:  Source: Mirror iOS Application The main menu of the Mirror Application displays on-demand and live classes that are each "a video." The "Live" section of the Mirror Application main menu displays a preview of ongoing and upcoming live digital content including videos. The "Classes" section of the Mirror Application main menu displays a preview of on-demand digital including videos. Selecting a class causes the digital content to stream over the Internet and playback on the Mirror Application. Selecting a class causes the Mirror Application to provide options to stream the class to a variety of apparatuses over the Internet, including the iOS device that the Mirror Application is executing on or the separate Mirror Device.

USP 10,757,156 to Mirror

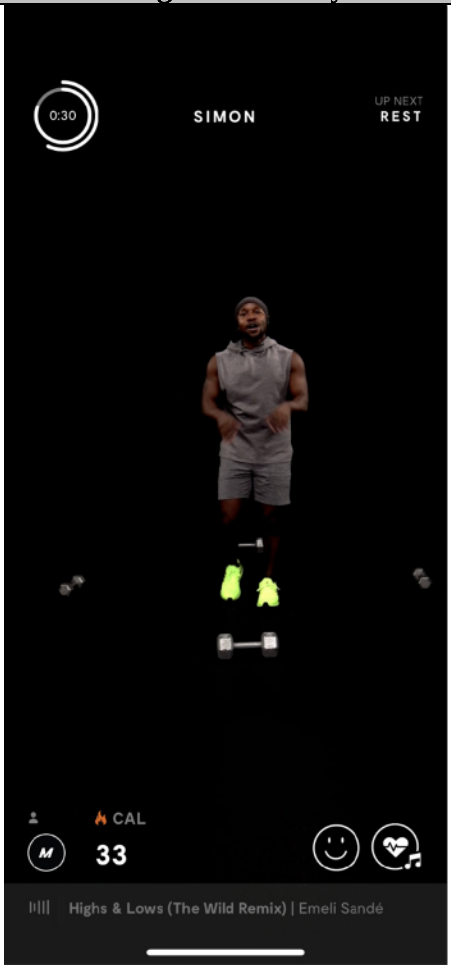
Claim	Claim Limitation	Infringement Analysis
		 Source: Mirror iOS Application

USP 10,757,156 to Mirror

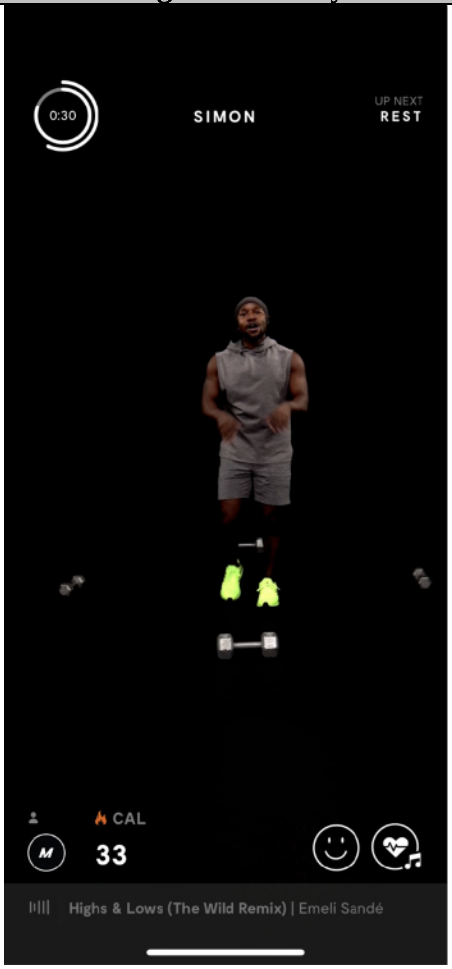
Claim	Claim Limitation	Infringement Analysis
		 Source: Mirror iOS Application Selecting “Your Mirror” causes the digital content to be streamed on the user’s Mirror device, which is connected to the Internet via TCP/IP protocols.

USP 10,757,156 to Mirror

Claim	Claim Limitation	Infringement Analysis
		 Alternatively, selecting "This Device" causes the digital content to be streamed on the user's iOS device:

Claim	Claim Limitation	Infringement Analysis
		 Source: Mirror iOS Application As set forth above, Mirror Devices are "for rendering a video that is adaptively received as a digital stream from a video server over a network." The Mirror Devices obtain streams of selected digital content for adaptive-rate content streaming. The streams are obtained over a network, specifically the Internet using TCP/IP protocols.

Claim	Claim Limitation	Infringement Analysis
		As described in greater detail below, the Mirror Application and Mirror devices perform adaptive-rate streaming of such digital video streams such that the rendered video is "adaptively received" from a video server.
	a media player operating on the apparatus, wherein the media player is configured to stream the video from the video server via at least one transmission control protocol (TCP) connection over the network,	The Mirror Application includes "a media player" that operates on the apparatus running the Mirror Application. When digital content such as a class is selected, the Mirror Application launches a media player that "configured to stream a video from the video server via at least one transmission control protocol (TCP) connection over the network," as shown below.

Claim	Claim Limitation	Infringement Analysis
		 Source: Mirror iOS Application The Mirror Devices include a "media player" that "operat[es] on the" Mirror Device "apparatus." For example, when the "Your Mirror" option is selected in the Mirror Application after selecting a class, the Mirror Device launches a media player that is "configured to stream a video from the video server via at least one transmission control protocol (TCP) connection over the network," as shown below.

USP 10,757,156 to Mirror

Claim	Claim Limitation	Infringement Analysis
		 As shown in greater detail below, the digital content is streamed.
	wherein the video server stores multiple	Multiple different copies of the video encoded at different bit rates as multiple sets of streamlets are stored.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Infringement Analysis</th></tr>
<tr><td></td><td>different copies of the video encoded at different bit rates as multiple sets of streamlets,</td><td>
For the following test, a live video was selected. In the test, an iPhone 11 running the Mirror Application makes an HTTPS GET request for a master playlist named “playlist.m3u8” that specifies the “multiple different copies of the video encoded at different bitrates” and provides links to the playlists for the multiple different copies of the video. In response to the request, the following master playlist file named “playlist.m3u8” is returned.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080
../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720
../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480
../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360
../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288
../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180
../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8

```

Filename: playlist.m3u8.

This is a master playlist file according to the HLS specification.[1] The playlist shows six different copies of the video, denoted by each #EXT-X-STREAM-INF tag at the following bandwidths:

- 6434112 (referred to herein as “6434112 Bandwidth”)
- 864048 (referred to herein as “864048 Bandwidth”)
- 403824 (referred to herein as “403824 Bandwidth”)
- 367728 (referred to herein as “367728 Bandwidth”)
- 312832 (referred to herein as “312832 Bandwidth”)
</td></tr>
</table>

[1] RFC 8216 (HLS Live Streaming), Section 4.3.4 (Master Playlist Tags)

Claim	Claim Limitation	Infringement Analysis
		• 249664 (referred to herein as "**249664 Bandwidth**") For each of the copies, the master playlist provides a link to a playlist file for the specified copy of the selected video at a particular bandwidth and resolution, which is called a "variant" in HLS. Each of the playlist files for each of the different copies of the video further include links to segments or streamlets of the video for the respective bandwidth and resolution of the copy. For example, the Mirror Application issued a request for the variant playlist file corresponding to the **6434112 Bandwidth** copy of the video, which is named "**chunklist.m3u8**." That file, including the links to the streamlets associated with that copy, is shown below.

USP 10,757,156 to Mirror

Claim	Claim Limitation	Infringement Analysis
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1232 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:08.356Z #EXTINF:2.0, r4vhrugx/00000000/media_1232.ts #EXTINF:2.0, r4vhrugx/00000000/media_1233.ts #EXTINF:2.0, r4vhrugx/00000000/media_1234.ts #EXTINF:2.0, r4vhrugx/00000000/media_1235.ts #EXTINF:2.0, r4vhrugx/00000000/media_1236.ts #EXTINF:2.0, r4vhrugx/00000000/media_1237.ts #EXTINF:2.0, r4vhrugx/00000000/media_1238.ts #EXTINF:2.0, r4vhrugx/00000000/media_1239.ts #EXTINF:2.0, r4vhrugx/00000000/media_1240.ts #EXTINF:2.0, r4vhrugx/00000000/media_1241.ts #EXTINF:2.0, Filename: https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 The variant playlist file corresponding to the **403824 Bandwidth** copy of the video, titled "**chunklist.m3u8**," is shown below.

Claim	Claim Limitation	Infringement Analysis
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1238 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z #EXTINF:2.0, fbd862nq/00000000/media_1238.ts #EXTINF:2.0, fbd862nq/00000000/media_1239.ts #EXTINF:2.0, fbd862nq/00000000/media_1240.ts #EXTINF:2.0, fbd862nq/00000000/media_1241.ts #EXTINF:2.0, fbd862nq/00000000/media_1242.ts #EXTINF:2.0, fbd862nq/00000000/media_1243.ts #EXTINF:2.0, fbd862nq/00000000/media_1244.ts #EXTINF:2.0, fbd862nq/00000000/media_1245.ts #EXTINF:2.0, fbd862nq/00000000/media_1246.ts #EXTINF:2.0, fbd862nq/00000000/media_1247.ts #EXTINF:2.0, fbd862nq/00000000/media_1248.ts Filename: **chunklist.m3u8** Each of the segments or streamlets of the set corresponding to each of the different copies of the video are also stored, as shown by the Mirror Application issuing a GET request for "media_1238.ts." The various versions of the segments with different bandwidths and different resolutions are stored and accessed based on requests from the Mirror Application and Mirror Devices. The **6434112**

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Infringement Analysis</th></tr>
<tr><td></td><td></td><td>Bandwidth set of segments of the program are encoded and stored in a directory and the 403824 Bandwidth set of segments of the program are encoded and stored in a directory. As explained above, these versions have different bit rates which are identified in the master playlist file (i.e., filename: playlist.m3u8). The identical segments of the filenames in each directory demonstrate that these segments are copies of the same video.</td></tr>
<tr><td></td><td>wherein each of the streamlets yields a different portion of the video on playback,</td><td>Each of the .ts files (e.g., segments or streamlets) “yields a different portion of the video on playback.” An excerpt of the Charles Proxy sequence listing is provided below and shows the Mirror Application requesting and receiving different, sequential 2 second segments of the program to playback different portions of the video. The sequence listing below shows the Mirror Application requesting and receiving three sequential segments of the program “media_1274.ts,” “media_1275.ts,” and “media_1276.ts.” The separate files are separate segments for different time indexes and portions of the video on playback. As discussed above, there are multiple versions of each of these files for each time index.

<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1274.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1275.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1276.ts</td></tr>
</table>

On information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above.</td></tr>
<tr><td></td><td>wherein the streamlets across the different copies yield the same portions of the video on playback,</td><td>The “streamlets across the different copies yield the same portions of the video on playback” on the Mirror Application and Mirror Devices. As described above, each of the playlists includes links to the files with the same video content at different bandwidths and resolutions.

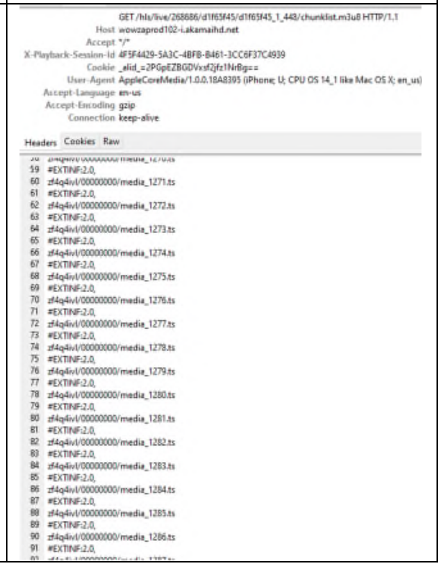
For example, each variant playlist includes multiple streamlets, including a streamlet with the filename ending in “media_1274.ts” A comparison of the 6434112 Bandwidth, 403824 Bandwidth, and</td></tr>
</table>

Claim	Claim Limitation	Infringement Analysis
		249664 Bandwidth copies from above shows that each of the playlists includes the "**media_1274.ts**" segment. On information and belief, playlists for the other copies also include this segment. As discussed above, each streamlet corresponds to a portion of the video on playback. Notably, each bitrate copy of the **media_1274.ts** segment has a duration of 2 seconds (as noted in each line beginning with #EXTINF and corresponds to the same time index, thereby yielding "the same portions of the video on playback." Upon information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application.
	and wherein the streamlets in the different copies are aligned in time such that the streamlets that play back the same portion of the video for the different copies each begin at the same playback time in relation to the beginning of the video,	The "streamlets in the different copies are aligned in time such that the streamlets that play back the same portion of the video for the different copies each begin at the same playback time in relation to the beginning of the video." For example, compare the segment files in the **6434112 Bandwidth**, **403823 Bandwidth**, and **249664 Bandwidth** copies of the video:

USP 10,757,156 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Infringement Analysis</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>6434112 Bandwidth</th><th>403824 Bandwidth</th><th>249664 Bandwidth</th></tr>
<tr><td>

<pre>
GET /hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 HTTP/1.1
Host wowzaprod102-i.akamaihd.net
Accept */*
X-Playback-Session-Id 4F5F4429-5A3C-4BFB-B461-3CC6F37C4939
Cookie _alid_=2PGpEZBGDVxsf2jfz1NrBg==
User-Agent AppleCoreMedia/1.0.0.18A8395 (iPhone; U; CPU OS 14_1 like Mac OS X; en_us)
Accept-Language en-us
Accept-Encoding gzip
Connection keep-alive
Headers Cookies Raw
70 r4vhrugx/00000000/media_1263.ts
71 #EXTINF:2.0,
72 r4vhrugx/00000000/media_1264.ts
73 #EXTINF:2.0,
74 r4vhrugx/00000000/media_1265.ts
75 #EXTINF:2.0,
76 r4vhrugx/00000000/media_1266.ts
77 #EXTINF:2.0,
78 r4vhrugx/00000000/media_1267.ts
79 #EXTINF:2.0,
80 r4vhrugx/00000000/media_1268.ts
81 #EXTINF:2.0,
82 r4vhrugx/00000000/media_1269.ts
83 #EXTINF:2.0,
84 r4vhrugx/00000000/media_1270.ts
85 #EXTINF:2.0,
86 r4vhrugx/00000000/media_1271.ts
87 #EXTINF:2.0,
88 r4vhrugx/00000000/media_1272.ts
89 #EXTINF:2.0,
90 r4vhrugx/00000000/media_1273.ts
91 #EXTINF:2.0,
92 r4vhrugx/00000000/media_1274.ts
93 #EXTINF:2.0,
94 r4vhrugx/00000000/media_1275.ts
95 #EXTINF:2.0,
96 r4vhrugx/00000000/media_1276.ts
97 #EXTINF:2.0,
98 r4vhrugx/00000000/media_1277.ts
99 #EXTINF:2.0,
100 r4vhrugx/00000000/media_1278.ts
101 #EXTINF:2.0,
102 r4vhrugx/00000000/media_1279.ts
</pre>
</td><td>

<pre>
GET /hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 HTTP/1.1
Host wowzaprod102-i.akamaihd.net
Accept */*
X-Playback-Session-Id 4F5F4429-5A3C-4BFB-B461-3CC6F37C4939
Cookie _alid_=2PGpEZBGDVxsf2jfz1NrBg==
User-Agent AppleCoreMedia/1.0.0.18A8395 (iPhone; U; CPU OS 14_1 like Mac OS X; en_us)
Accept-Language en-us
Accept-Encoding gzip
Connection keep-alive
Headers Cookies Raw
70 fbd862nq/00000000/media_1269.ts
71 #EXTINF:2.0,
72 fbd862nq/00000000/media_1270.ts
73 #EXTINF:2.0,
74 fbd862nq/00000000/media_1271.ts
75 #EXTINF:2.0,
76 fbd862nq/00000000/media_1272.ts
77 #EXTINF:2.0,
78 fbd862nq/00000000/media_1273.ts
79 #EXTINF:2.0,
80 fbd862nq/00000000/media_1274.ts
81 #EXTINF:2.0,
82 fbd862nq/00000000/media_1275.ts
83 #EXTINF:2.0,
84 fbd862nq/00000000/media_1276.ts
85 #EXTINF:2.0,
86 fbd862nq/00000000/media_1277.ts
87 #EXTINF:2.0,
88 fbd862nq/00000000/media_1278.ts
89 #EXTINF:2.0,
90 fbd862nq/00000000/media_1279.ts
91 #EXTINF:2.0,
92 fbd862nq/00000000/media_1280.ts
93 #EXTINF:2.0,
94 fbd862nq/00000000/media_1281.ts
95 #EXTINF:2.0,
96 fbd862nq/00000000/media_1282.ts
97 #EXTINF:2.0,
98 fbd862nq/00000000/media_1283.ts
99 #EXTINF:2.0,
100 fbd862nq/00000000/media_1284.ts
101 #EXTINF:2.0,
102 fbd862nq/00000000/media_1285.ts
</pre>
</td><td>

<pre>
GET /hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 HTTP/1.1
Host wowzaprod102-i.akamaihd.net
Accept */*
X-Playback-Session-Id 4F5F4429-5A3C-4BFB-B461-3CC6F37C4939
Cookie _alid_=2PGpEZBGDVxsf2jfz1NrBg==
User-Agent AppleCoreMedia/1.0.0.18A8395 (iPhone; U; CPU OS 14_1 like Mac OS X; en_us)
Accept-Language en-us
Accept-Encoding gzip
Connection keep-alive
Headers Cookies Raw
59 #EXTINF:2.0,
60 zf4q4ivl/00000000/media_1271.ts
61 #EXTINF:2.0,
62 zf4q4ivl/00000000/media_1272.ts
63 #EXTINF:2.0,
64 zf4q4ivl/00000000/media_1273.ts
65 #EXTINF:2.0,
66 zf4q4ivl/00000000/media_1274.ts
67 #EXTINF:2.0,
68 zf4q4ivl/00000000/media_1275.ts
69 #EXTINF:2.0,
70 zf4q4ivl/00000000/media_1276.ts
71 #EXTINF:2.0,
72 zf4q4ivl/00000000/media_1277.ts
73 #EXTINF:2.0,
74 zf4q4ivl/00000000/media_1278.ts
75 #EXTINF:2.0,
76 zf4q4ivl/00000000/media_1279.ts
77 #EXTINF:2.0,
78 zf4q4ivl/00000000/media_1280.ts
79 #EXTINF:2.0,
80 zf4q4ivl/00000000/media_1281.ts
81 #EXTINF:2.0,
82 zf4q4ivl/00000000/media_1282.ts
83 #EXTINF:2.0,
84 zf4q4ivl/00000000/media_1283.ts
85 #EXTINF:2.0,
86 zf4q4ivl/00000000/media_1284.ts
87 #EXTINF:2.0,
88 zf4q4ivl/00000000/media_1285.ts
89 #EXTINF:2.0,
90 zf4q4ivl/00000000/media_1286.ts
91 #EXTINF:2.0,
</pre>
</td></tr>
</table>
These three playlists refer to segments with identical filename conventions ("media_#.ts") and identical lengths (in seconds) with different file path prefixes. In particular, for example, file names containing "media_1275.ts" are all 2.0 seconds in length and are available for both bandwidths (and corresponding resolutions and bit rates). Thus, the segments with similar file paths and names, for example "r4vhrugx/00000000/media_1271.ts," "fbd862nq/00000000/media_1271.ts," and "zf4qivl/00000000/media_1271.ts" are streamlets of different copies that are aligned in time such that they playback the same portion of the video included in the "1271" segment and beginning at the same playback time in relation to the beginning of the video.
</td></tr>
<tr><td></td><td>and wherein the media player streams the video by: requesting sequential streamlets of one of</td><td>The media player of the Mirror Application streams the video by "requesting sequential streamlets of one of the copies from the video server according to the playback times of the streamlets by transmitting hypertext transport protocol (HTTP) GET requests that identify the selected streamlets stored by the video server."</td></tr>
</table>

USP 10,757,156 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Infringement Analysis</th></tr>
<tr><td></td><td>the copies from the video server according to the playback times of the streamlets by transmitting hypertext transport protocol (HTTP) GET requests that identify the selected streamlets stored by the video server,</td><td>The sequence listing below, captured using the Charles Proxy application, shows the Mirror Application requesting and receiving three sequential segments of the 249664 Bandwidth copy of the video: “media_1274.ts,” “media_1275.ts,” and “media_1276.ts.” The files are sequential and separate segments for different time indexes. As discussed above, there are multiple versions of each of these files for each time index.

<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1274.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1275.ts</td></tr>
<tr><td>GET</td><td>wowzaprod11-2-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1-448/zf4q4ivl/00000000/media_1276.ts</td></tr>
</table>

As shown above, the sequential streamlets of one of the copies (i.e., the 249664 Bandwidth copy) are requested by transmitting HTTP GET requests. Further, the sequential streamlets are requested according to the playback times and the requests identify the selected streamlets as shown for example in the ‘path” column of the sequence chart shown above. As the next 2.0 second streamlet is needed, it is requested by the Mirror Application.

On information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application, as shown above.</td></tr>
<tr><td></td><td>wherein the sequential streamlets are selected by the media player from the based upon successive determinations to shift the playback quality to a higher or lower quality one of</td><td>The Mirror Application media player selects the sequential streamlets, such as those shown above, based upon successive determinations to shift the playback quality to a higher or lower quality one of the different copies of the video.

In order to demonstrate the Mirror Application’s selection and determinations to shift the playback quality, the throttling feature of the Charles Proxy application was used to limit the Mirror Application’s bandwidth to approximate a slower speed, and then the throttling setting was removed.

When the bandwidth for the Mirror Application is reduced, the Mirror Application engages in adaptation to shift playback quality to a lower quality one of the different copies by requesting a lower</td></tr>
</table>

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Infringement Analysis</th></tr>
<tr><td></td><td>the different copies of the video;</td><td>bit rate version of the content at a subsequent time index, and when bandwidth for the Mirror Application is unconstrained, the Mirror Application engages in adaptation to shift playback quality to a higher quality one of the different copies in a similar way. This behavior demonstrates the automatic selection and requesting of streamlets dependent upon successive determinations by the media player to shift the playback quality. Recall from the discussion about the playlist.m3u8 file that the variant playlists and segments are stored in different directories, based upon the resolution and bandwidth.

<table>
<tr><th>Bandwidth</th><th>Playlist Filename</th></tr>
<tr><td>6434112</td><td>d1f65f45_1_4128/chunklist.m3u8</td></tr>
<tr><td>864048</td><td>d1f65f45_1_2728/chunklist.m3u8</td></tr>
<tr><td>403824</td><td>d1f65f45_1_1728/chunklist.m3u8</td></tr>
<tr><td>367728</td><td>d1f65f45_1_1152/chunklist.m3u8</td></tr>
<tr><td>312832</td><td>d1f65f45_1_640/chunklist.m3u8</td></tr>
<tr><td>249664</td><td>d1f65f45_1_448/chunklist.m3u8</td></tr>
</table>

The chosen resolutions of the Mirror Application can be determined based on which playlist and associated segments of the video are retrieved. A portion of the Charles Proxy sequence listing shows the requested and retrieved files while bandwidth was constrained and after the bandwidth was unconstrained is shown below.

<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1277.ts</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8</td></tr>
<tr><td colspan="3">…</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_640/3fyynadk/00000000/media_1278.ts</td></tr>
</table>
</td></tr>
</table>

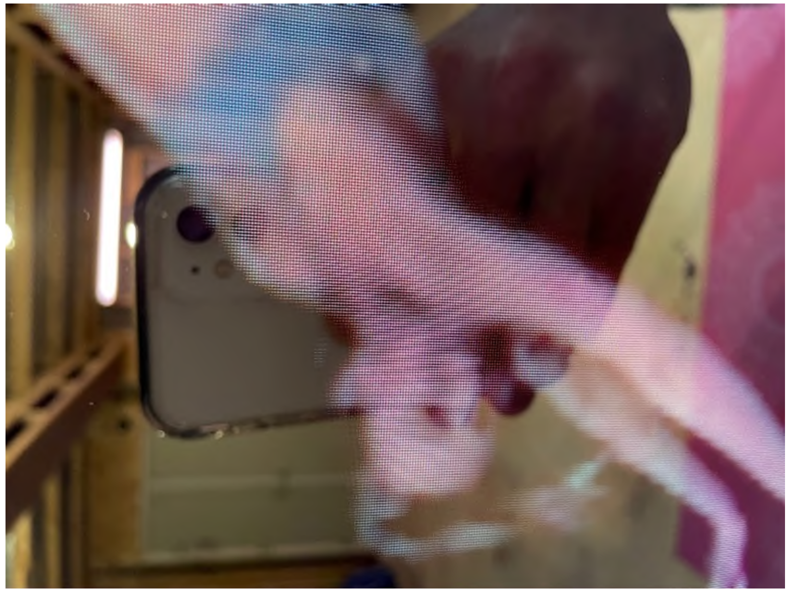
Claim	Claim Limitation	Infringement Analysis
		As shown above, when the Mirror Application operates at an unconstrained bandwidth, the Mirror Application transitions from a lower bitrate version of the video (e.g., the **249664 Bandwidth** version) to a higher bitrate version of the video (e.g., the **312832 Bandwidth** version). Between these two segments, the Mirror Application requests and receives a variant playlist file for the **312832 Bandwidth** version of the video ("/hls/live/268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8"). Also as demonstrated above, the Mirror Application requests and receives the lower resolution encoded files when the bandwidth is constrained and requests and receives the higher resolution encoded files when the bandwidth is unconstrained. In this way, the Mirror Application adapts subsequent segment requests based on successive determinations to shift the playback to a higher or lower quality one of the copies. On information and belief, the Mirror Devices operate in the same or substantially the same way, as shown above. For example, during a test of the Mirror Devices, the Mirror Devices successively determined to and shifted from a higher quality copy to a lower quality copy of the video when the bandwidth was constrained and then shifted back to a higher quality copy from the lower quality copy when the bandwidth was unconstrained, as shown below by the test capturing the shifting through multiple different copies of the video at varying qualities.

USP 10,757,156 to Mirror

Claim	Claim Limitation	Infringement Analysis
		First version: 

Claim	Claim Limitation	Infringement Analysis
		Second version: 

USP 10,757,156 to Mirror

Claim	Claim Limitation	Infringement Analysis
		Third version: 
	repeatedly generating, by the media player, a factor relating to the performance of the network that is indicative of an ability to sustain the streaming of the video;	The media player of the Mirror Application and Mirror Device repeatedly generates a factor relating to the performance of the network that is indicative of an ability to sustain the streaming of the video. The repeated generation of a factor relating to the performance of the network that is indicative of the ability to sustain the streaming of the video using the files from different ones of the copies is demonstrated by testing where bandwidth available to the Mirror Application and Mirror Devices is throttled and unthrottled and the playback quality automatically shifted in accordance with the player's ability to sustain the stream under the bandwidth constraints, as was shown above. Upon information and belief, the Mirror Devices operate in the same or substantially the same way. In addition, the Mirror Devices run the Android operating system. ExoPlayer is a ubiquitous HLS video player for Android. ExoPlayer2 provide a DefaultBandwidthMeter class that "[e]stimates bandwidth by listening to data transfers."

Claim	Claim Limitation	Infringement Analysis
		https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/upstream/DefaultBandwidthMeter.html. The ExoPlayer documentation further explains that "[t]he bandwidth estimate is calculated using a SlidingPercentile and is updated each time a transfer ends. The initial estimate is based on the current operator's network country code or the locale of the user, as well as the network connection type. This can be configured in the DefaultBandwidthMeter.Builder." *Id*. ExoPlayer's DefaultBandwidthMeter therefore repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the set of one or more factors relate to the performance of the network. The estimated bandwidth from the DefaultBandwidthMeter is then used by the AdaptiveTrackSelection.Factory class to determine whether to change to a higher or lower version of the stream. https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/trackselection/AdaptiveTrackSelection.Factory.html The AdaptiveTrackSelection.Factory class uses the following parameters to determine whether to switch: minDurationForQualityIncreaseMs - The minimum duration of buffered data required for the selected track to switch to one of higher quality. maxDurationForQualityDecreaseMs - The maximum duration of buffered data required for the selected track to switch to one of lower quality. minDurationToRetainAfterDiscardMs - When switching to a track of significantly higher quality, the selection may indicate that media already buffered at the lower quality can be discarded to speed up the switch. This is the minimum duration of media that must be retained at the lower quality.

Claim	Claim Limitation	Infringement Analysis
		bandwidthFraction - The fraction of the available bandwidth that the selection should consider available for use. Setting to a value less than 1 is recommended to account for inaccuracies in the bandwidth estimator. *Id*.
	adapting the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the streamlets of the highest quality copy of the video that is determined to be sustainable at that time; and	The Mirror Application adapts the successive determinations to shift the playback quality based on the factor to achieve continuous playback of the video using the streamlets of the highest quality copy of the video that is determined to be sustainable at that time. As noted above, the Mirror Application and Mirror Devices shift between playback quality based on a factor that includes, for example, bandwidth limitations to enable continuous playback of the video. Further, the Mirror Application and Mirror Devices request a lower quality content when bandwidth is constrained, as shown herein. Accordingly, the Mirror Application and Mirror Devices continue to request the highest quality content that is sustainable when doing so. When the Mirror Application operates at unconstrained bandwidth, the Mirror Application transitions from a lower-bitrate version of the video (**249664 Bandwidth** version) to a higher-bitrate version of the video (**367728 Bandwidth** version). Between these two successive segments, the Mirror Application requests and receives a **367728 Bandwidth** variant playlist file.

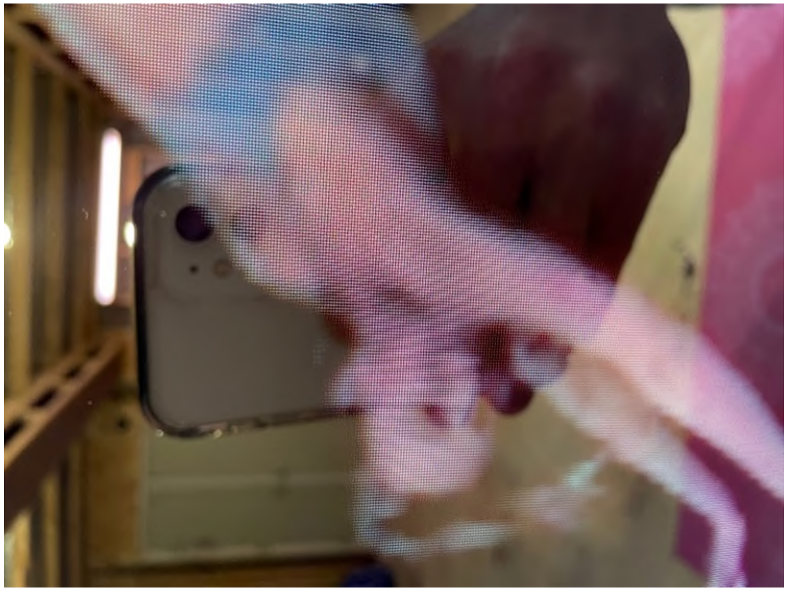
Method	Host	Path
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
..		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1279.ts
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1152/bmit701z/00000000/media_1278.ts

Claim	Claim Limitation	Infringement Analysis
		As demonstrated above, the Mirror Application requests and receives the higher resolution segment when operating at unconstrained bandwidth. A portion of the Charles Proxy sequence listing below shows the requested and retrieved files prior to and just after bandwidth was constrained is shown below. **Method** / **Host** / **Path**: GET / wowzaprod102-i.akamaihd.net / /hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 .. GET / wowzaprod102-i.akamaihd.net / /hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts … GET / wowzaprod102-i.akamaihd.net / /hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 … GET / wowzaprod102-i.akamaihd.net / /hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts When the Mirror Application operates at a constrained bandwidth, the Mirror Application transitions from a higher-bitrate version of the video (the **403824 Bandwidth** version) to a lower-bitrate version of the video (the **249664 Bandwidth** version). Between the two segments, the Mirror Application requests and receives a **249664 Bandwidth** variant playlist file of the video. As demonstrated above, the Mirror Application requests and receives the lower resolution encoded files while the bandwidth is constrained. In operation, when full bandwidth is returned, the Mirror Application again requests and plays a higher bitrate segment of the video, which is in response the factor being greater than a threshold. Upon information and belief, the Mirror Devices operates in the same or substantially the same way. For example, when bandwidth is unconstrained, the Mirror Devices shift to the highest quality copy of the video that is sustainable.

Method	Host	Path
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8
..		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
…		
GET	wowzaprod102-i.akamaihd.net	/hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts

Claim	Claim Limitation	Infringement Analysis
		First version:  For the current test, when the bandwidth was limited, the Mirror Device subsequently shifted to a lower quality copy of the video.

Claim	Claim Limitation	Infringement Analysis
		Second version:  When the bandwidth was constrained further, the Mirror Device subsequently shifted to another lower quality copy of the video.

Claim	Claim Limitation	Infringement Analysis
		Third version:  When the bandwidth constraint was removed, the Mirror Device subsequently returned to a higher-bandwidth version shown above. In addition, the Mirror Devices run the Android operating system. ExoPlayer is a ubiquitous HLS video player for Android. ExoPlayer2 provide a DefaultBandwidthMeter class that "[e]stimates bandwidth by listening to data transfers." https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/upstream/DefaultBandwidthMeter.html. The ExoPlayer documentation further explains that "[t]he bandwidth estimate is calculated using a SlidingPercentile and is updated each time a transfer ends. The initial estimate is based on the current operator's network country code or the locale of the user, as well as the network connection type. This can be configured in the DefaultBandwidthMeter.Builder." *Id*.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Infringement Analysis</th></tr>
<tr><td></td><td></td><td>ExoPlayer’s DefaultBandwidthMeter therefore repeatedly generating a factor indicative of the current ability to sustain the streaming of the video using the files from different ones of the copies, wherein the set of one or more factors relate to the performance of the network.

The estimated bandwidth from the DefaultBandwidthMeter is then used by the AdaptiveTrackSelection.Factory class to determine whether to change to a higher or lower version of the stream. https://exoplayer.dev/doc/reference/com/google/android/exoplayer2/trackselection/AdaptiveTrackSelection.Factory.html The AdaptiveTrackSelection.Factory class uses the following parameters to determine whether to switch:

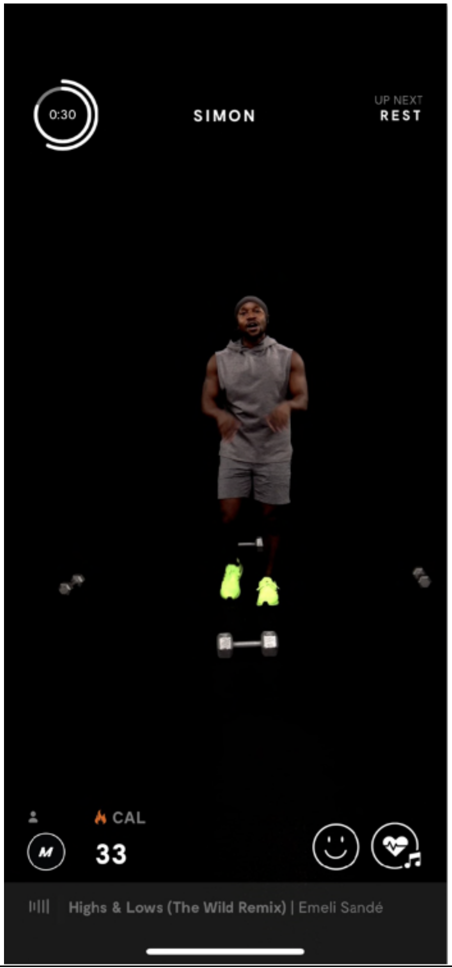
minDurationForQualityIncreaseMs - The minimum duration of buffered data required for the selected track to switch to one of higher quality.

maxDurationForQualityDecreaseMs - The maximum duration of buffered data required for the selected track to switch to one of lower quality.

minDurationToRetainAfterDiscardMs - When switching to a track of significantly higher quality, the selection may indicate that media already buffered at the lower quality can be discarded to speed up the switch. This is the minimum duration of media that must be retained at the lower quality.

bandwidthFraction - The fraction of the available bandwidth that the selection should consider available for use. Setting to a value less than 1 is recommended to account for inaccuracies in the bandwidth estimator.

Id.</td></tr>
<tr><td></td><td>presenting the video for playback by providing the requested streamlets</td><td>As shown above, the Mirror Application and Mirror Devices receive the playlist file that lists the .ts file segments in order of ascending playback time and the Mirror Application requests those same .ts files in order of ascending playback time. The Mirror Application and Mirror Devices then present the video by playing back the requested media files with the media player on the apparatus in order of</td></tr>
</table>

Claim	Claim Limitation	Infringement Analysis
	in order of ascending start time.	ascending playback time. The Mirror Application and Mirror Devices playback the requested .ts files in order of ascending playback time after they are retrieved. For example, the media player of the Mirror Application includes a timer or timeline indicating that the media files are played back in order of ascending playback time in accordance with the timeline. 

Claim	Claim Limitation	Infringement Analysis
		For example, the media player of the Mirror Devices includes a timer or timeline indicating that the media files are played back in order of ascending playback time in accordance with the timeline. 

EXHIBIT E

US010951680B2

(12) United States Patent
Brueck et al.

(10) Patent No.: US 10,951,680 B2
(45) Date of Patent: *Mar. 16, 2021

(54) APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING

(71) Applicant: **DISH Technologies L.L.C.**, Englewood, CO (US)

(72) Inventors: **David F. Brueck**, Saratoga Springs, UT (US); **Mark B. Hurst**, Cedar Hills, UT (US); **R. Drew Major**, Orem, UT (US)

(73) Assignee: **DISH Technologies L.L.C.**, Englewood, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **16/876,604**

(22) Filed: **May 18, 2020**

(65) **Prior Publication Data**

US 2020/0280595 A1 Sep. 3, 2020

Related U.S. Application Data

(63) Continuation of application No. 16/004,056, filed on Jun. 8, 2018, now Pat. No. 10,659,513, which is a

(Continued)

(51) **Int. Cl.**

H04L 29/06 (2006.01)
H04L 12/927 (2013.01)

(Continued)

(52) **U.S. Cl.**

CPC ***H04L 65/607*** (2013.01); ***G06F 16/183*** (2019.01); ***G06F 16/71*** (2019.01);

(Continued)

(58) **Field of Classification Search**

CPC H04N 19/34; H04N 19/40; H04N 21/234327; H04N 21/2662;

(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,535,355	A	8/1985	Am et al.
5,168,356	A	12/1992	Acampora et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CA	2466482	A1	5/2003
EP	0365683	A1	5/1990

(Continued)

OTHER PUBLICATIONS

Roy, S., et al., "Architecture of a Modular Streaming Media Server for Content Delivery Networks," 2002 IEEE. Published in the 2003 International Conference on Multimedia and Expo ICME 2001.

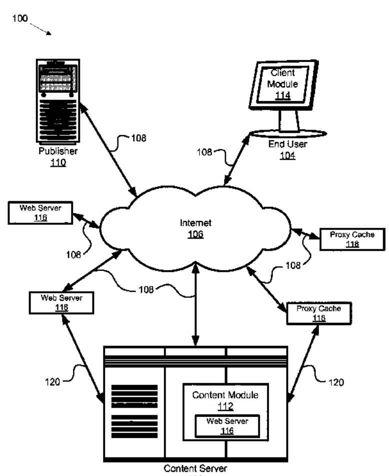
(Continued)

Primary Examiner — Chirag R Patel
(74) *Attorney, Agent, or Firm* — Lorenz & Kopf LLP

(57) ABSTRACT

An apparatus for multi-bitrate content streaming includes a receiving module configured to capture media content, a streamlet module configured to segment the media content and generate a plurality of streamlets, and an encoding module configured to generate a set of streamlets. The system includes the apparatus, wherein the set of streamlets comprises a plurality of streamlets having identical time indices and durations, and each streamlet of the set of streamlets having a unique bitrate, and wherein the encoding module comprises a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid. A method includes receiving media content, segmenting the media content and generating a plurality of streamlets, and generating a set of streamlets.

29 Claims, 11 Drawing Sheets



Related U.S. Application Data

continuation of application No. 15/414,025, filed on Jan. 24, 2017, now Pat. No. 9,998,516, which is a continuation of application No. 14/719,122, filed on May 21, 2015, now Pat. No. 9,571,551, which is a continuation of application No. 14/106,051, filed on Dec. 13, 2013, now Pat. No. 9,071,668, which is a continuation of application No. 13/617,114, filed on Sep. 14, 2012, now Pat. No. 8,612,624, which is a continuation of application No. 12/906,940, filed on Oct. 18, 2010, now Pat. No. 8,402,156, which is a continuation of application No. 11/673,483, filed on Feb. 9, 2007, now Pat. No. 7,818,444, which is a continuation-in-part of application No. 11/116,783, filed on Apr. 28, 2005, now Pat. No. 8,868,772.

(60) Provisional application No. 60/566,831, filed on Apr. 30, 2004.

(51) **Int. Cl.**

H04L 12/801	(2013.01)
G06F 16/71	(2019.01)
G06F 16/182	(2019.01)
H04N 7/24	(2011.01)
H04N 21/2343	(2011.01)
H04N 21/433	(2011.01)
H04N 21/84	(2011.01)
H04N 21/845	(2011.01)
H04L 29/08	(2006.01)
H04N 21/2662	(2011.01)

(52) **U.S. Cl.**

CPC ***H04L 29/06027*** (2013.01); ***H04L 47/12*** (2013.01); ***H04L 47/801*** (2013.01); ***H04L 65/1069*** (2013.01); ***H04L 65/4069*** (2013.01); ***H04L 65/608*** (2013.01); ***H04L 65/80*** (2013.01); ***H04L 67/02*** (2013.01); ***H04L 67/2842*** (2013.01); ***H04L 67/32*** (2013.01); ***H04N 7/24*** (2013.01); ***H04N 21/23439*** (2013.01); ***H04N 21/2662*** (2013.01); ***H04N 21/4331*** (2013.01); ***H04N 21/84*** (2013.01); ***H04N 21/8456*** (2013.01)

(58) **Field of Classification Search**

CPC . H04N 21/2393; H04L 65/80; H04L 67/2842; H04L 65/4069; H04L 65/607; H04L 65/608

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,267,334	A	11/1993	Normille et al.	
5,404,446	A	4/1995	Bowater et al.	
5,687,095	A	11/1997	Haskell et al.	
5,732,183	A	3/1998	Sugiyama	
5,768,527	A	6/1998	Zhu et al.	
5,812,786	A *	9/1998	Seazholtz	H04M 11/062 370/465
5,841,432	A	11/1998	Carmel et al.	
5,953,506	A	9/1999	Kalra et al.	
6,091,775	A	7/2000	Hibi et al.	
6,091,777	A	7/2000	Guetz et al.	
6,122,660	A	9/2000	Baransky et al.	
6,172,672	B1	1/2001	Ramasubramanian et al.	
6,185,736	B1	2/2001	Ueno	
6,195,680	B1	2/2001	Goldszmidt et al.	
6,366,614	B1	4/2002	Pian et al.	
6,374,289	B2	4/2002	Delaney et al.	
6,389,473	B1	5/2002	Carmel et al.	
6,449,719	B1	9/2002	Baker	
6,486,803	B1	11/2002	Luby et al.	
6,490,627	B1	12/2002	Kalra et al.	
6,510,553	B1	1/2003	Hazra	
6,574,591	B1	6/2003	Kleiman et al.	
6,604,118	B2	8/2003	Klleiman et al.	
6,618,752	B1	9/2003	Moore et al.	
6,708,213	B1	3/2004	Bommaiah et al.	
6,721,723	B1	4/2004	Gibson et al.	
6,731,600	B1	5/2004	Patel et al.	
6,757,796	B1	6/2004	Hofmann	
6,760,772	B2	7/2004	Zou et al.	
6,795,863	B1	9/2004	Doty, Jr.	
6,845,107	B1	1/2005	Kitazawa et al.	
6,850,965	B2	2/2005	Allen	
6,859,839	B1	2/2005	Zahorjan et al.	
6,874,015	B2	3/2005	Kaminsky et al.	
6,968,387	B2	11/2005	Lanphear	
6,976,090	B2	12/2005	Ben-Shaul et al.	
7,054,365	B2	5/2006	Kim et al.	
7,054,774	B2	5/2006	Batterberry et al.	
7,054,911	B1	5/2006	Lango et al.	
7,075,986	B2	7/2006	Girod et al.	
7,093,001	B2	8/2006	Yang et al.	
7,096,271	B1	8/2006	Omoigui et al.	
7,099,954	B2	8/2006	Li et al.	
7,116,894	B1	10/2006	Chatterton	
7,174,385	B2	2/2007	Li	
7,194,549	B1	3/2007	Lee et al.	
7,240,100	B1	7/2007	Wein et al.	
7,260,640	B1	8/2007	Kramer et al.	
7,274,740	B2	9/2007	van Beek et al.	
7,295,520	B2	11/2007	Lee et al.	
7,310,678	B2	12/2007	Gunaseelan et al.	
7,325,073	B2	1/2008	Shao et al.	
7,328,243	B2	2/2008	Yaeger et al.	
7,330,908	B2	2/2008	Jungek	
7,334,044	B1	2/2008	Allen	
7,349,358	B2	3/2008	Hennessey et al.	
7,349,976	B1	3/2008	Glaser et al.	
7,369,610	B2 *	5/2008	Xu	H04N 21/2662 375/240.08
7,376,747	B2	5/2008	Hartop	
7,391,717	B2	6/2008	Kiemets et al.	
7,408,984	B2	8/2008	Lu et al.	
7,412,531	B1	8/2008	Lango et al.	
7,477,688	B1	1/2009	Zhang et al.	
7,523,181	B2	4/2009	Swildens et al.	
7,536,469	B2	5/2009	Chou et al.	
7,546,355	B2	6/2009	Kalnitsky	
7,558,869	B2	7/2009	Leon et al.	
7,577,750	B2	8/2009	Shen et al.	
7,593,333	B2	9/2009	Li et al.	
7,599,307	B2	10/2009	Seckin et al.	
7,609,652	B2	10/2009	Kellerer et al.	
7,653,735	B2	1/2010	Mandato et al.	
7,707,303	B2	4/2010	Albers et al.	
7,719,985	B2	5/2010	Lee et al.	
7,760,801	B2	7/2010	Ghanbari et al.	
7,779,135	B2	8/2010	Hudson et al.	
7,788,395	B2	8/2010	Bowra et al.	
7,797,439	B2	9/2010	Cherkasova et al.	
7,817,985	B2	10/2010	Moon	
7,818,444	B2	10/2010	Brueck et al.	
7,925,781	B1	4/2011	Chan et al.	
7,934,159	B1 *	4/2011	Rahman	H04N 21/4825 715/716
8,036,265	B1	10/2011	Reynolds et al.	
8,370,514	B2	2/2013	Hurst et al.	
8,402,156	B2	3/2013	Brueck et al.	
8,521,836	B2	8/2013	Kewalramani et al.	
8,612,624	B2	12/2013	Brueck et al.	
8,683,066	B2	3/2014	Hurst et al.	
8,686,066	B2	4/2014	Kwampian et al.	
8,868,772	B2	10/2014	Major et al.	
8,880,721	B2	11/2014	Hurst et al.	
9,344,496	B2	5/2016	Hurst et al.	
9,462,074	B2	10/2016	Guo et al.	
2001/0013128	A1	8/2001	Hagai et al.	

(56) **References Cited**

U.S. PATENT DOCUMENTS

2001/0047423 A1 11/2001 Shao et al.
2002/0029274 A1 3/2002 Allen
2002/0073167 A1 6/2002 Powell et al.
2002/0091840 A1 7/2002 Pulier et al.
2002/0097750 A1 7/2002 Gunaseelan et al.
2002/0131496 A1 9/2002 Vasudevan et al.
2002/0144276 A1 10/2002 Radford et al.
2002/0152317 A1 10/2002 Wang et al.
2002/0152318 A1 10/2002 Menon et al.
2002/0156912 A1 10/2002 Hurst et al.
2002/0161898 A1 10/2002 Hartop et al.
2002/0161908 A1 10/2002 Benitez et al.
2002/0161911 A1 10/2002 Pinckney, III et al.
2002/0169926 A1 11/2002 Pinckney, III et al.
2002/0174434 A1 11/2002 Lee et al.
2002/0176418 A1 11/2002 Hunt et al.
2002/0178330 A1 11/2002 Schlowsky-Fischer et al.
2002/0188745 A1 12/2002 Hughes et al.
2003/0005455 A1 1/2003 Bowers
2003/0014684 A1 1/2003 Kashyap
2003/0018966 A1 1/2003 Cook et al.
2003/0021166 A1 1/2003 Soloff
2003/0021282 A1 1/2003 Hospodor
2003/0023982 A1 * 1/2003 Lee H04N 21/234363 725/116
2003/0055995 A1 3/2003 Ala Honkola
2003/0065803 A1 4/2003 Heuvelman
2003/0067872 A1 4/2003 Harrell et al.
2003/0081582 A1 5/2003 Jain et al.
2003/0093790 A1 5/2003 Logan et al.
2003/0103571 A1 * 6/2003 Meehan H04N 19/34 375/240.27
2003/0107994 A1 6/2003 Jacobs et al.
2003/0135631 A1 7/2003 Li et al.
2003/0135863 A1 7/2003 VanDer Schaar
2003/0140159 A1 7/2003 Campbell et al.
2003/0151753 A1 8/2003 Li et al.
2003/0152036 A1 8/2003 Quigg Brown et al.
2003/0154239 A1 8/2003 Davis et al.
2003/0195977 A1 10/2003 Liu et al.
2003/0204519 A1 10/2003 Sirivara et al.
2003/0204602 A1 10/2003 Hudson et al.
2003/0233464 A1 12/2003 Walpole et al.
2003/0236904 A1 12/2003 Walpole et al.
2004/0003101 A1 1/2004 Roth et al.
2004/0010613 A1 1/2004 Apostolopoulos et al.
2004/0030547 A1 2/2004 Leaning et al.
2004/0030599 A1 2/2004 Sie et al.
2004/0030797 A1 2/2004 Akinlar et al.
2004/0031054 A1 2/2004 Dankworth et al.
2004/0049780 A1 3/2004 Gee
2004/0054551 A1 3/2004 Ausubel et al.
2004/0071209 A1 4/2004 Burg et al.
2004/0083283 A1 4/2004 Sundaram et al.
2004/0093420 A1 5/2004 Gamble
2004/0103444 A1 5/2004 Weinberg et al.
2004/0117427 A1 6/2004 Allen et al.
2004/0143672 A1 7/2004 Padmanabham et al.
2004/0168052 A1 8/2004 Clisham et al.
2004/0170392 A1 9/2004 Lu et al.
2004/0179032 A1 9/2004 Huang
2004/0199655 A1 10/2004 Davies et al.
2004/0220926 A1 11/2004 Lamkin et al.
2004/0221088 A1 11/2004 Lisitsa et al.
2004/0260701 A1 12/2004 Lehikoinen et al.
2004/0267956 A1 12/2004 Leon et al.
2005/0015509 A1 1/2005 Sitaraman
2005/0033855 A1 2/2005 Moradi et al.
2005/0055425 A1 * 3/2005 Lango H04N 7/17318 709/219
2005/0066063 A1 3/2005 Grigorovitch et al.
2005/0076136 A1 4/2005 Cho et al.
2005/0084166 A1 4/2005 Bonch et al.
2005/0108414 A1 5/2005 Taylor et al.
2005/0120107 A1 6/2005 Kagan et al.
2005/0123058 A1 6/2005 Greenbaum et al.
2005/0185578 A1 8/2005 Padmanabham et al.
2005/0188051 A1 8/2005 Sneh
2005/0204046 A1 9/2005 Watanabe
2005/0251832 A1 11/2005 Chiueh
2005/0262257 A1 11/2005 Major et al.
2006/0010003 A1 1/2006 Kruse
2006/0059223 A1 3/2006 Klemets et al.
2006/0075446 A1 4/2006 Klemets et al.
2006/0080718 A1 4/2006 Gray et al.
2006/0130118 A1 6/2006 Damm
2006/0133809 A1 6/2006 Chow et al.
2006/0165166 A1 7/2006 Chou et al.
2006/0168290 A1 7/2006 Doron
2006/0168295 A1 7/2006 Batterberry et al.
2006/0206246 A1 9/2006 Walker
2006/0236219 A1 10/2006 Grigorovitch et al.
2006/0277564 A1 12/2006 Jarman
2007/0024705 A1 2/2007 Richter et al.
2007/0030833 A1 2/2007 Pirzada et al.
2007/0067480 A1 3/2007 Beek et al.
2007/0079325 A1 4/2007 de Heer
2007/0094405 A1 4/2007 Zhang
2007/0204310 A1 8/2007 Hua et al.
2007/0280255 A1 12/2007 Tsang et al.
2008/0028428 A1 1/2008 Jeong et al.
2008/0037527 A1 2/2008 Chan et al.
2008/0046939 A1 2/2008 Lu et al.
2008/0056373 A1 3/2008 Newlin et al.
2008/0104647 A1 5/2008 Hannuksela
2008/0120330 A1 5/2008 Reed et al.
2008/0120342 A1 5/2008 Reed et al.
2008/0133766 A1 6/2008 Luo
2008/0162713 A1 7/2008 Bowra et al.
2008/0184688 A1 8/2008 Daly et al.
2008/0195744 A1 8/2008 Bowra et al.
2008/0205291 A1 8/2008 Li et al.
2008/0219151 A1 9/2008 Ma et al.
2008/0222235 A1 9/2008 Hurst et al.
2008/0263180 A1 10/2008 Hurst et al.
2008/0281803 A1 11/2008 Gentric
2009/0043906 A1 2/2009 Hurst et al.
2009/0055471 A1 2/2009 Kozat et al.
2009/0055547 A1 2/2009 Hudson et al.
2009/0210549 A1 8/2009 Hudson et al.
2010/0098103 A1 4/2010 Xiong et al.
2010/0262711 A1 10/2010 Bouazizi
2011/0307545 A1 12/2011 Bouazizi
2015/0058496 A1 2/2015 Hurst et al.

FOREIGN PATENT DOCUMENTS

EP 0919952 A1 6/1999
EP 1202487 A2 5/2002
EP 1298931 A2 4/2003
EP 1395014 A1 3/2004
EP 1670256 A2 6/2006
EP 1777969 4/2007
GB 2367219 A 3/2002
JP 2000-201343 7/2000
JP 200192752 4/2001
JP 2011004225 A 1/2011
WO 2001067264 A1 9/2001
WO 2004025405 A2 3/2004
WO 2006010113 A2 1/2006

OTHER PUBLICATIONS

Bommaiah, E., et al., "Design and Implementation of a Caching System for Streaming Media over the Internet," 2000 IEEE. Published in RTAS '00 Proceedings of the Sixth IEEE Real Time Technology and Applications Symposium (RTAS 2000), p. 111.

Defendant JADOO TV, Inc.'s Disclosure of Invalidity Contentions, U.S. N. Dist. Ca. Case No. 5:18-cv-05214-EJD dated Sep. 22, 2020.

Defendant JADOO TV, Inc.'s Disclosure of Invalidity Contentions Appendix A, U.S. N. Dist. Ca. Case No. 5:18-cv-05214-EJD dated Sep. 22, 2020.

(56) **References Cited**

OTHER PUBLICATIONS

Balk et al., Adaptive Video Streaming: Pre-Encoded MPEG-4 with Bandwidth Scaling, 44 Computer Networks 415 (Mar. 2004).
Fujisawa, Hiroshi et al. "Implementaton of Efficient Access Mechanism for Multiple Mirror-Servers" IPSJ SIG Technical Report, vol. 2004, No. 9 (2004-DPS-116), Jan. 30, 2004, Information Processing Society of Japan, pp. 37-42.
Liu, Jiangchuan et al. "Adaptive Video Multicast Over the Internet" IEEE Computer Society, 2003.
"The meaning of performance factor—English-Japanese Weblio Dictionary", [online], Feb. 24, 2012, [searched on Feb. 24, 2012], the Internet <URL:http://ejje.weblio.jp/content/performance+factor>.
Tsuru, et al. "Recent evolution of the Internet measurement and inference techniques", IEICE Technical Report, vol. 103, No. 123, pp. 37-42, Jun. 12, 2003.
Rejaie, Reza et al. "Architectural Considerations for Playback of Quality Adaptive Video OVer the Internet" University of Southern California, Information Sciences Institute, 1998.
Roy, Sumit et al. "A System Architecture for Managing Mobile Streaming Media Services" Streaming Media Systems Group, Hewlett-Packard Laboratories, 2003.
Xu, Dongyan et al. "On Peer-to-Peer Media Streaming" Department of Computer Sciences, Purdue University, 2002.
Kozamerink, Franc "Media Streaming Over the Internet—An Over of Delivery Technologies" EBU Technical Review, Oct. 2002.
Lienhart, Rainer et al. "Challenges in Distributed Video Management and Delivery" Intel Corporation, EECS Dept., UC Berkeley, 2000-2002.
Zhang, Xinyan et al. "CoolStreaming/DONet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming" IEEE 2005.
Guo, Yang "DirectStream: A Directory-Based Peer-to-Peer Video Streaming Service" LexisNexis, Elsevier B.V. 2007.
Krasic et al., Quality-Adaptive Media Streaming by Priority Drop, Oregon Graduate Institute, 2001.
Krasic et al., QoS Scalability for Streamed Media Delivery, Oregon Graduate Institute School of Science & Engineering Technical Report CSE 99-011, Sep. 1999.
Huang et al., Adaptive Live Video Streaming by Priority Drop, Portland State University PDXScholar, Jul. 21, 2003.
Walpole et al, A Player for Adapctive MPEG Video Streaming Over the Internet, Oregon Graduate Institute of Science and Technology, Oct. 25, 2012.
Albanese, Andrew et al. "Priority Encoding Transmission", TR-94-039, Aug. 1994, 36 pgs, International Computer Science Institute, Berkeley, CA.
Birney, Bill "Intelligent Streaming", May 2003, Microsoft.
Goyal, Vivek K. "Multiple Description Coding: Compression Meets the Network," Sep. 2001, pp. 74-93, IEEE Signal Processing Magazine.
ON2 Technologies, Inc. "TrueMotion VP7 Video Codec" White Paper, Document Version 1.0, Jan. 10, 2005.
Pathan, Al-Mukaddim et al. "A Taxonomy and Survey of Content Delivery Networks" Australia, Feb. 2007, available at http://www.gridbus.org/reports/CDN-Taxonomy.pdf.
Puri, Rohit et al. "Multiple Description Source Coding Using Forward Error Correction Codes," Oct. 1999, 5 pgs., Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA.
Wicker, Stephen B. "Error Control Systems for Digital Communication and Storage," Prentice-Hall, Inc., New Jersey, USA, 1995, parts 1-6.
Liu, Jiangchuan et al. "Opportunities and Challenged of Peer-to-Peer Internet Video Broadcast," School of Computing Science, Simon Fraser University, British Columbia, Canada.
Clement, B. "Move Networks closes $11.3 Million on First Round VC Funding," Page One PR, Move Networks, Inc. Press Releases, Feb. 7, 2007, http://www.move.tv/press/press20070201.html.
Move Networks, Inc. "The Next Generation Video Publishing System," Apr. 11, 2007; http://www.movenetworks.com/wp-content/uploads/move-networks-publishing-system.pdf.
Yoshimura, Takeshi et al. "Mobile Streaming Media CDN Enabled by Dynamic SMIL", NTT DoCoMo, Multimedia Laboratories and Hewlett-Packard Laboratories,dated May 7-11, 2002, ACM 1-58113-449-5/02/0005; http://www2002.org/CDROM/refereed/515/.
Nguyen, T. et al., Multiple Sender Distributed Video Streaming, IEEE Transactinos on Multimedia, IEEE Service Center, Piscataway, NJ, US, vol. 6, No. 2, Apr. 1, 2004, pp. 315-326, XP011109142, ISSN: 1520-9210, DOI: 10.1109/TMM,2003.822790.

* cited by examiner

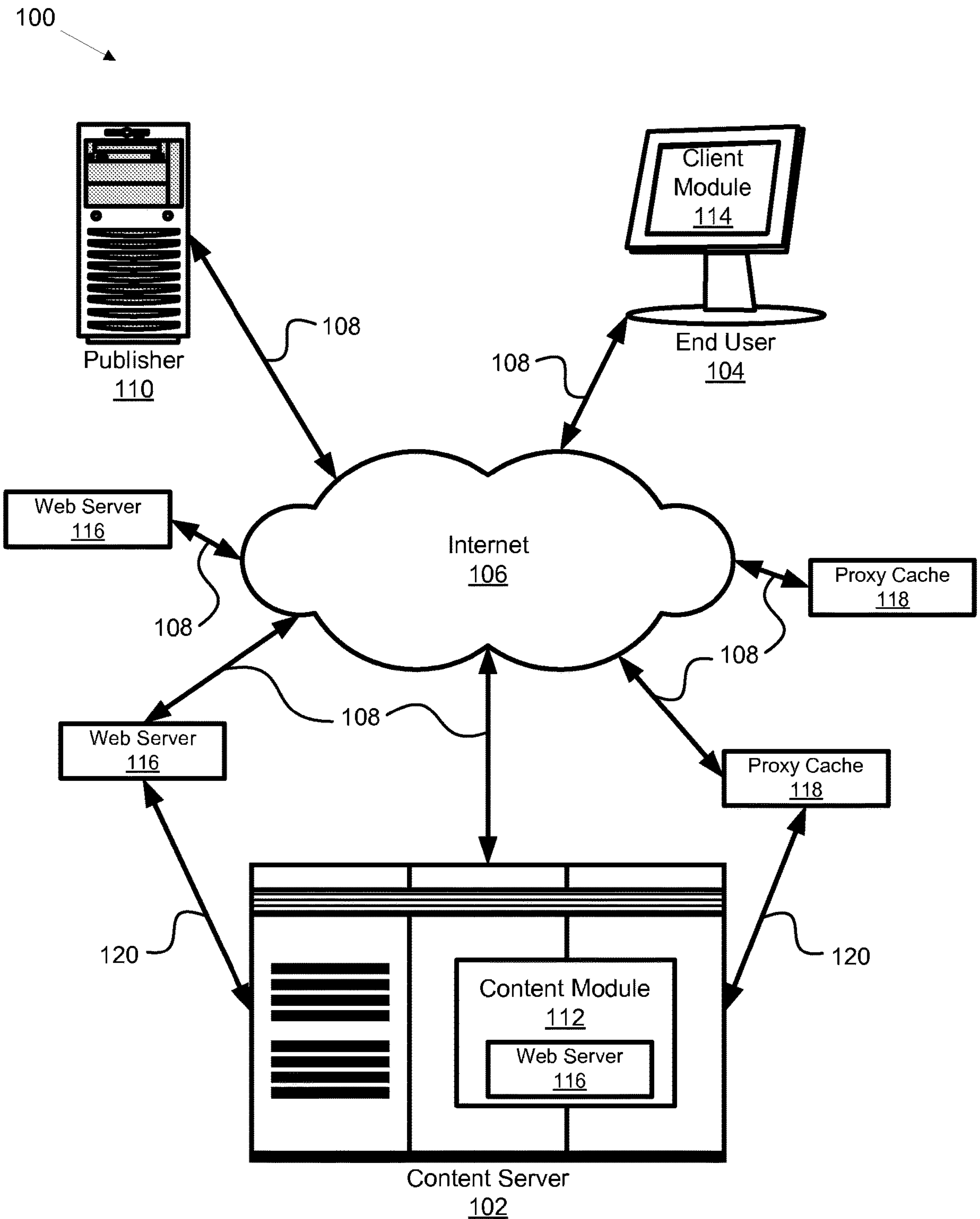

100
Client Module 114
Publisher 110
108
108
End User 104
Web Server 116
Internet 106
Proxy Cache 118
108
108
108
Web Server 116
Proxy Cache 118
120
Content Module 112
Web Server 116
120
Content Server 102

FIG. 1

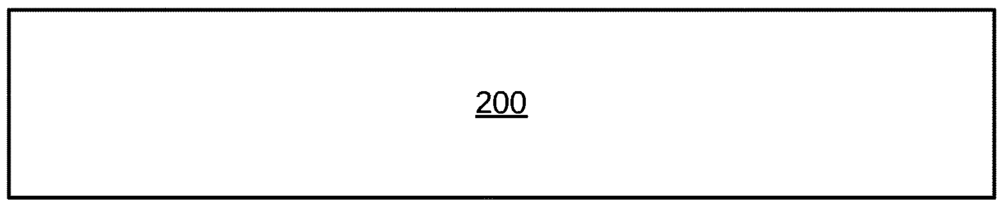


FIG. 2a

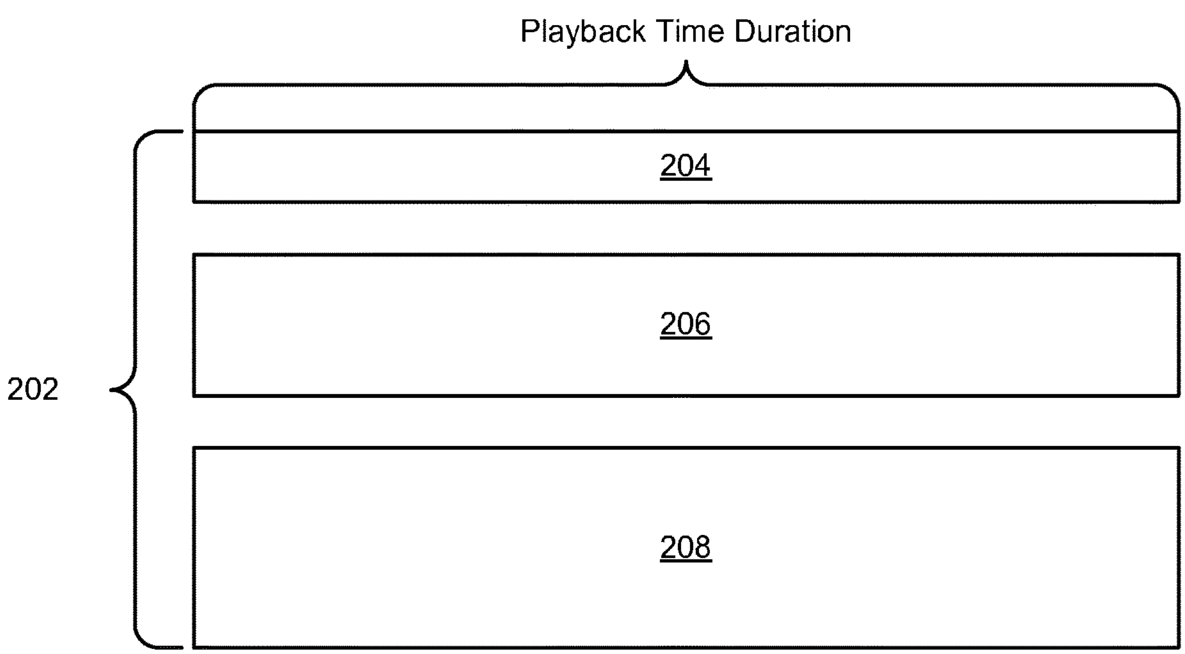


FIG. 2b

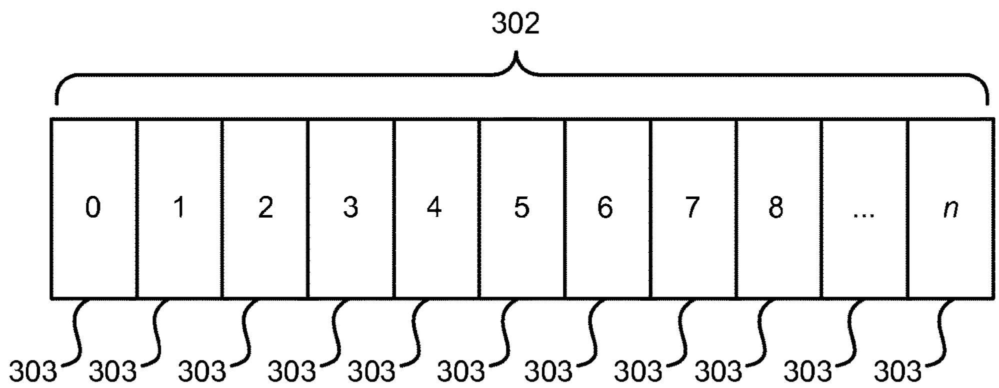

302
0
1
2
3
4
5
6
7
8
...
n
303
303
303
303
303
303
303
303
303
303
303

FIG. 3a

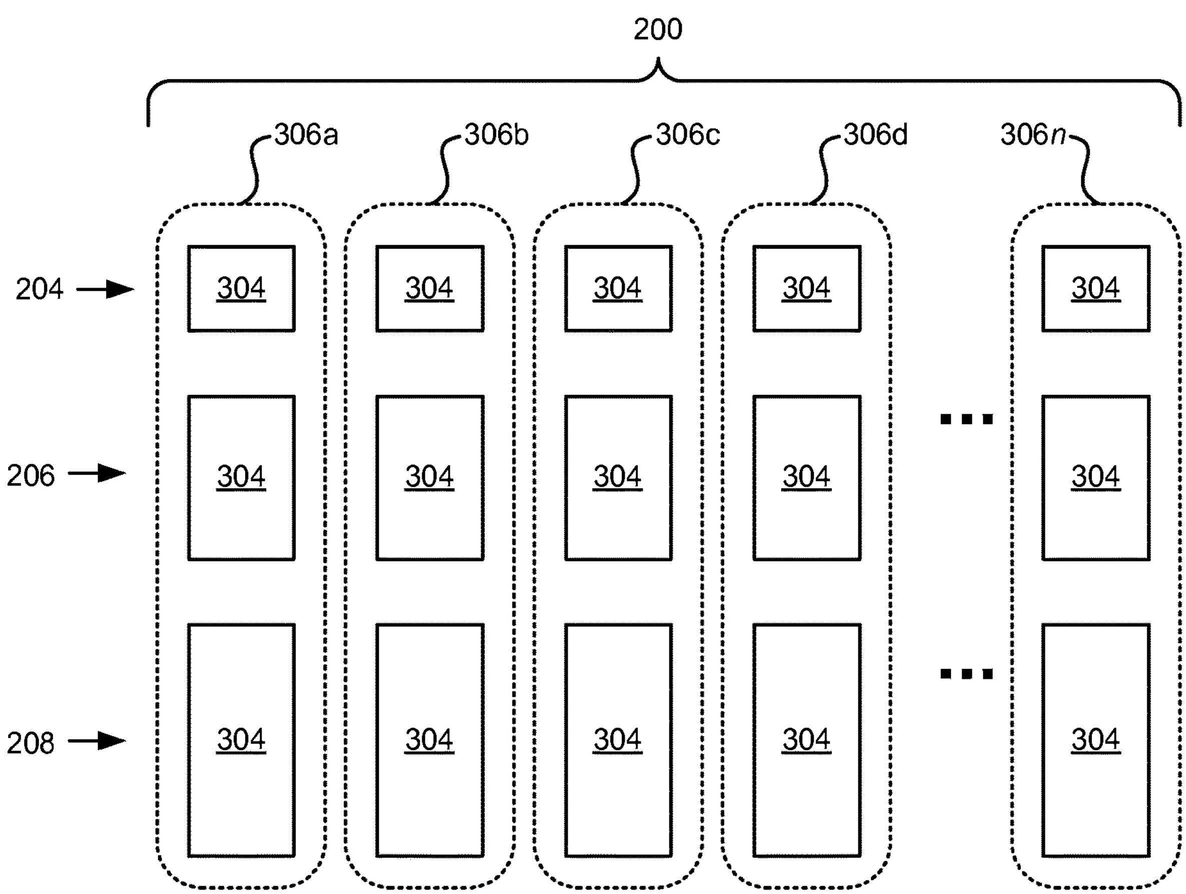

200
306a
306b
306c
306d
306n
204
206
208
304
304
304
304
304
304
304
304
304
304
304
304
304
304
304

FIG. 3b

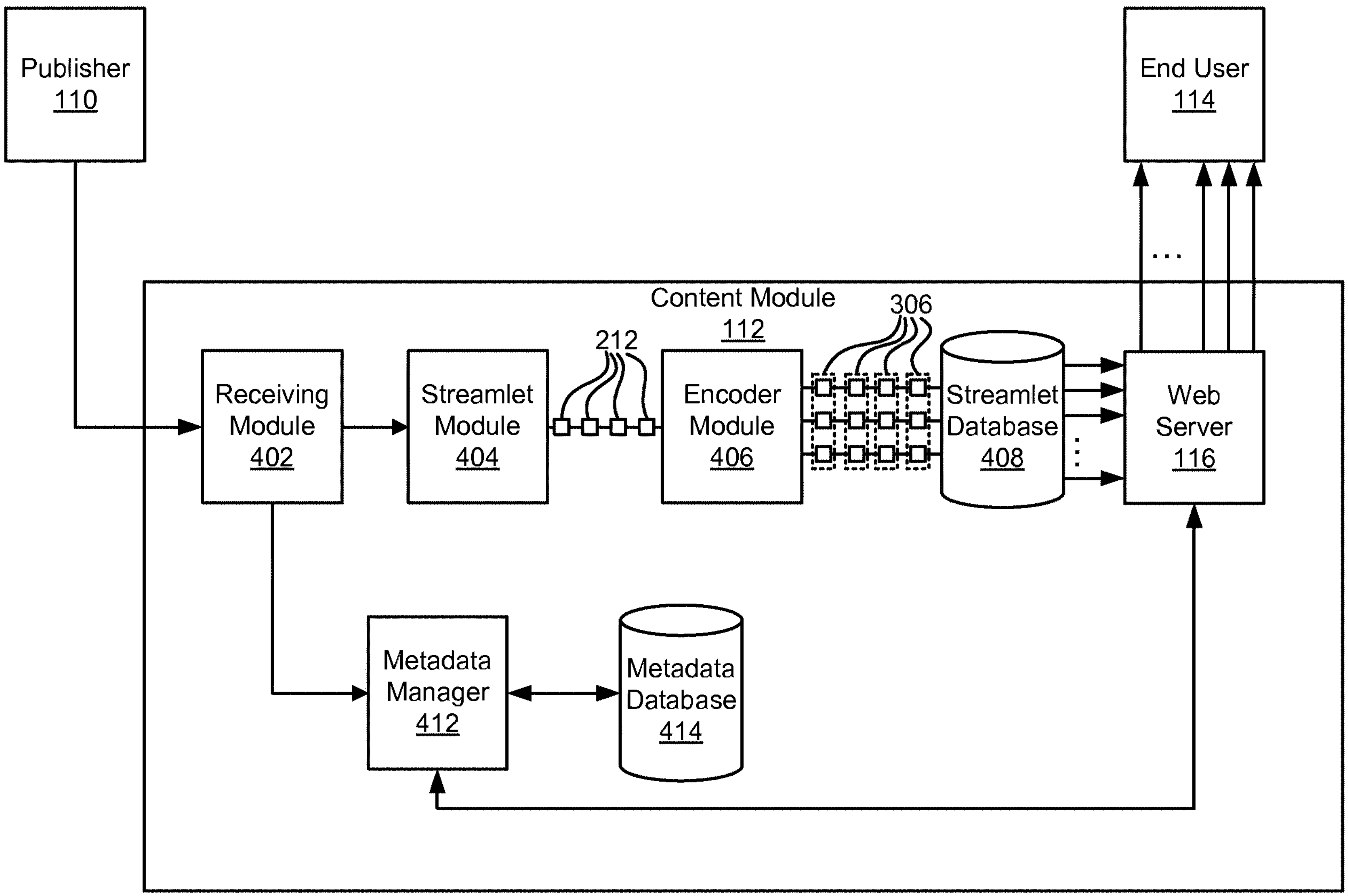

Publisher 110
End User 114
...
Content Module 112
306
212
Receiving Module 402
Streamlet Module 404
Encoder Module 406
Streamlet Database 408
Web Server 116
Metadata Manager 412
Metadata Database 414

FIG. 4

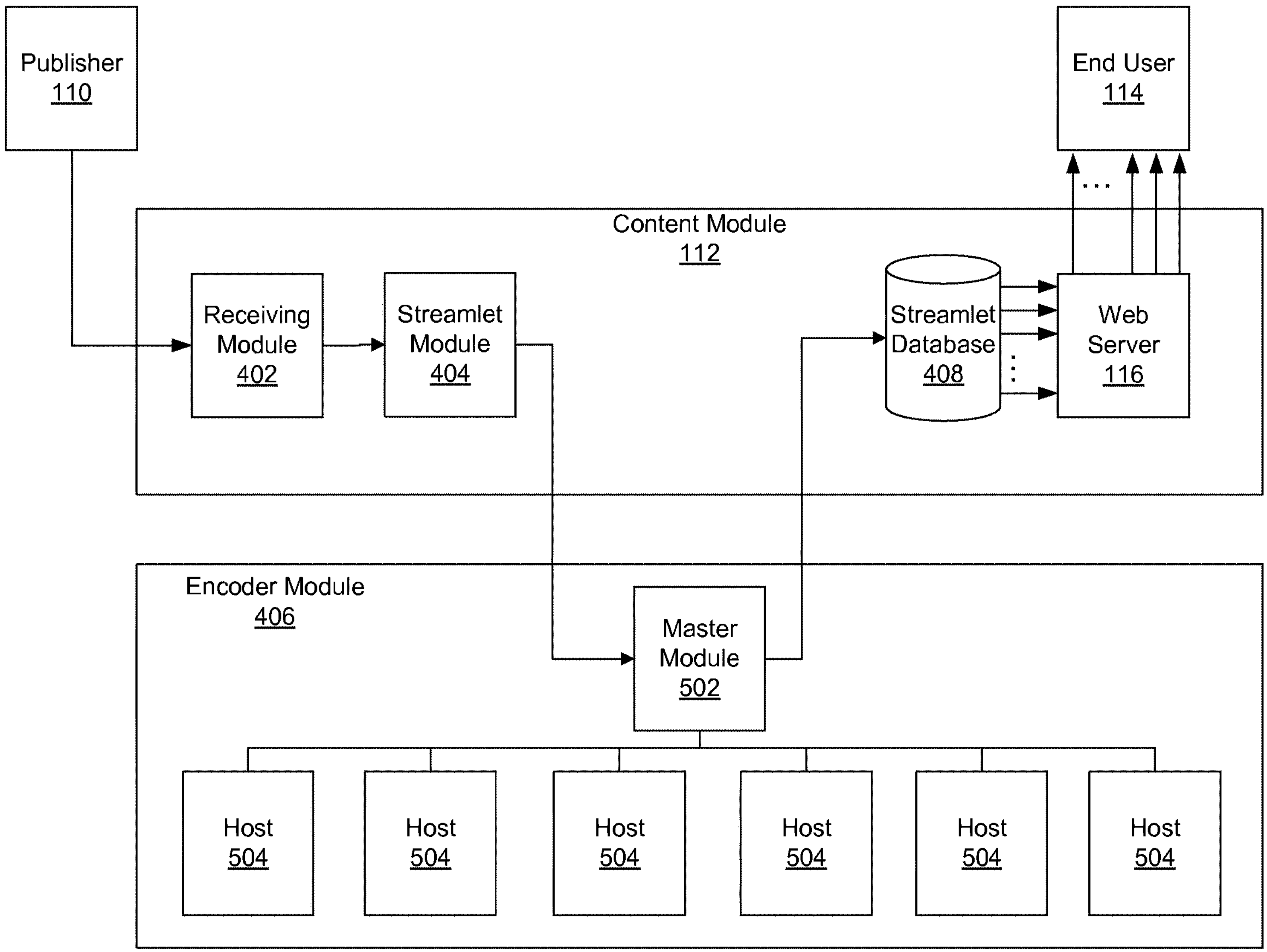

Publisher
110
End User
114
Content Module
112
Receiving Module
402
Streamlet Module
404
Streamlet Database
408
Web Server
116
Encoder Module
406
Master Module
502
Host
504
Host
504
Host
504
Host
504
Host
504
Host
504

FIG. 5a

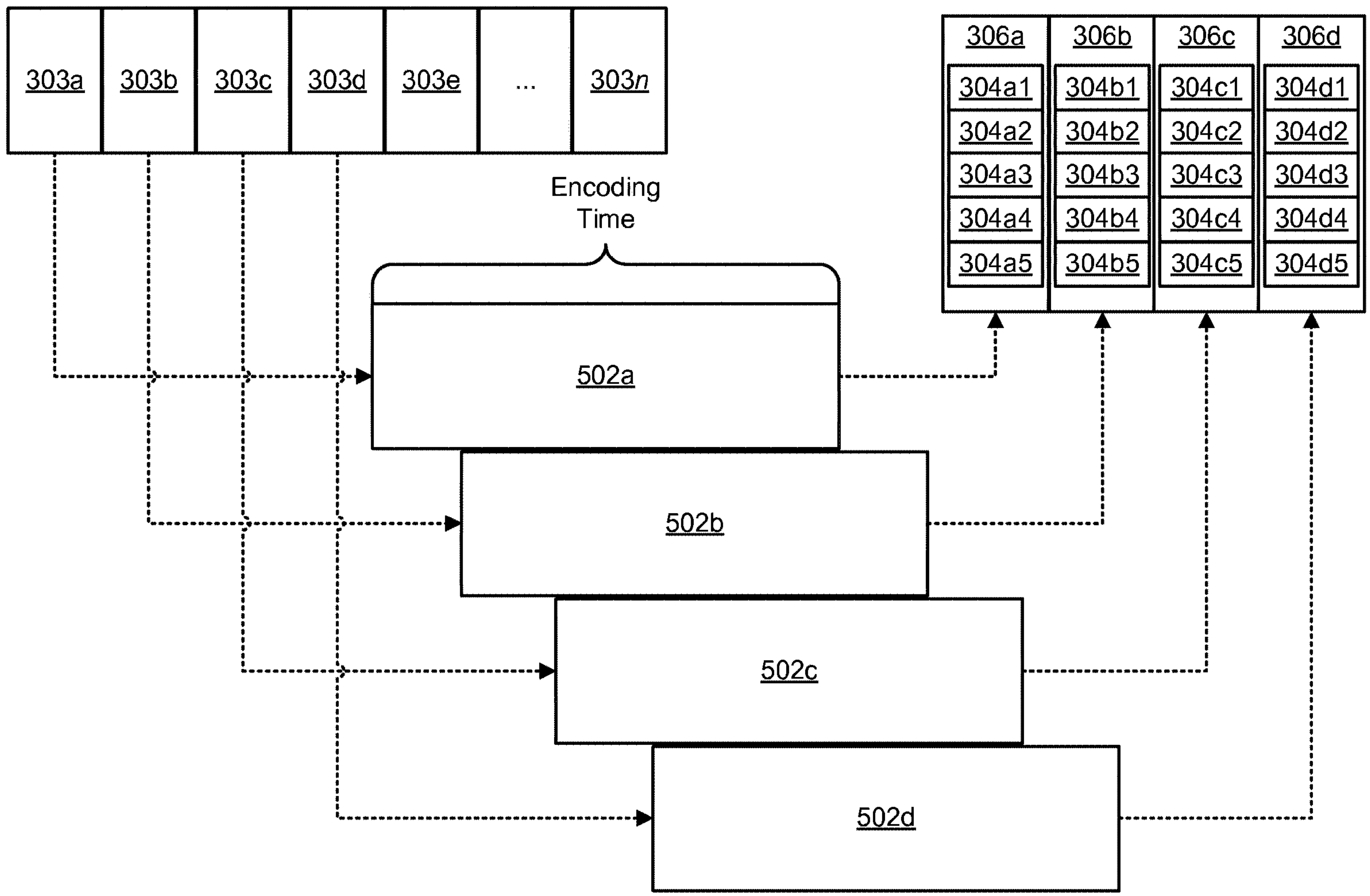

303a
303b
303c
303d
303e
...
303*n*
Encoding Time
502a
502b
502c
502d
306a
306b
306c
306d
304a1
304a2
304a3
304a4
304a5
304b1
304b2
304b3
304b4
304b5
304c1
304c2
304c3
304c4
304c5
304d1
304d2
304d3
304d4
304d5

FIG. 5b

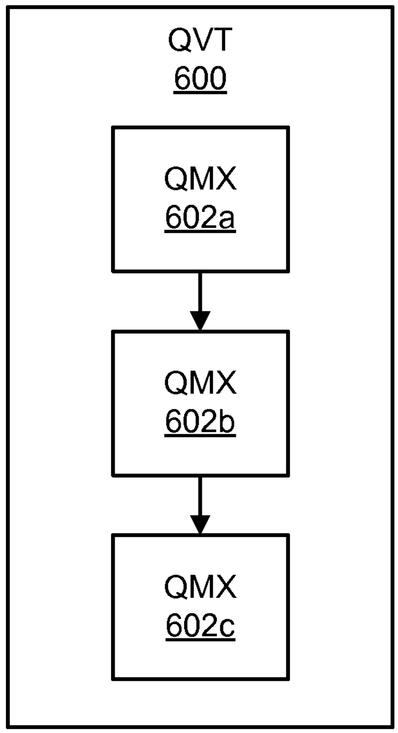


FIG. 6a

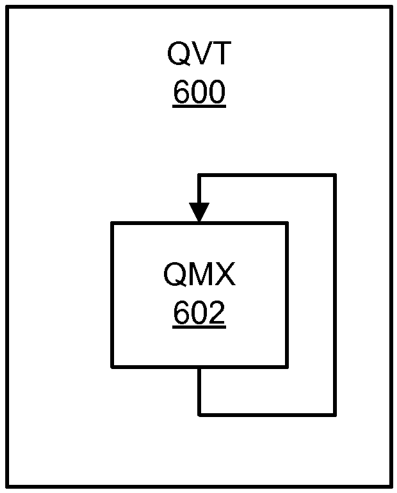


FIG. 6b

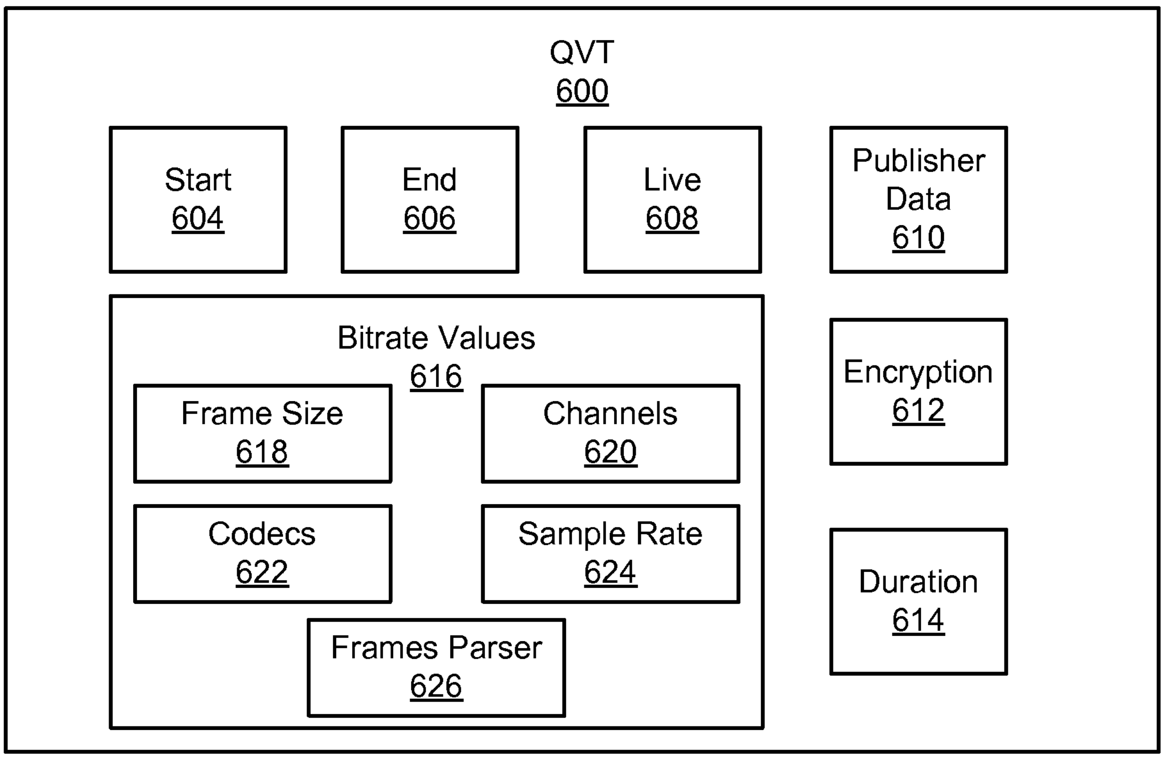


FIG. 6c

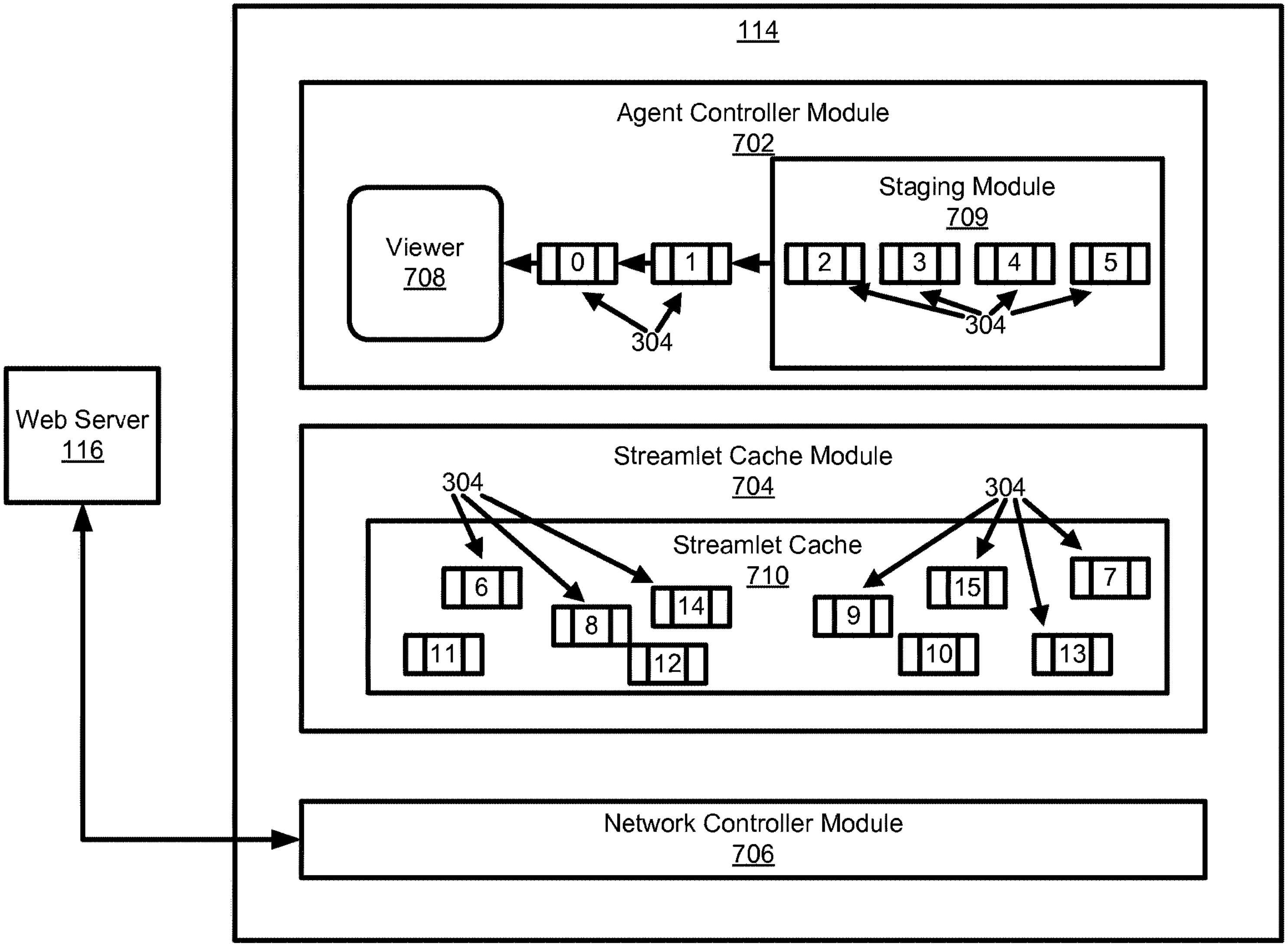

114
Agent Controller Module
702
Viewer
708
0
1
Staging Module
709
2
3
4
5
304
304
Web Server
116
Streamlet Cache Module
704
304
304
Streamlet Cache
710
6
8
14
11
12
9
15
7
10
13
Network Controller Module
706

FIG. 7

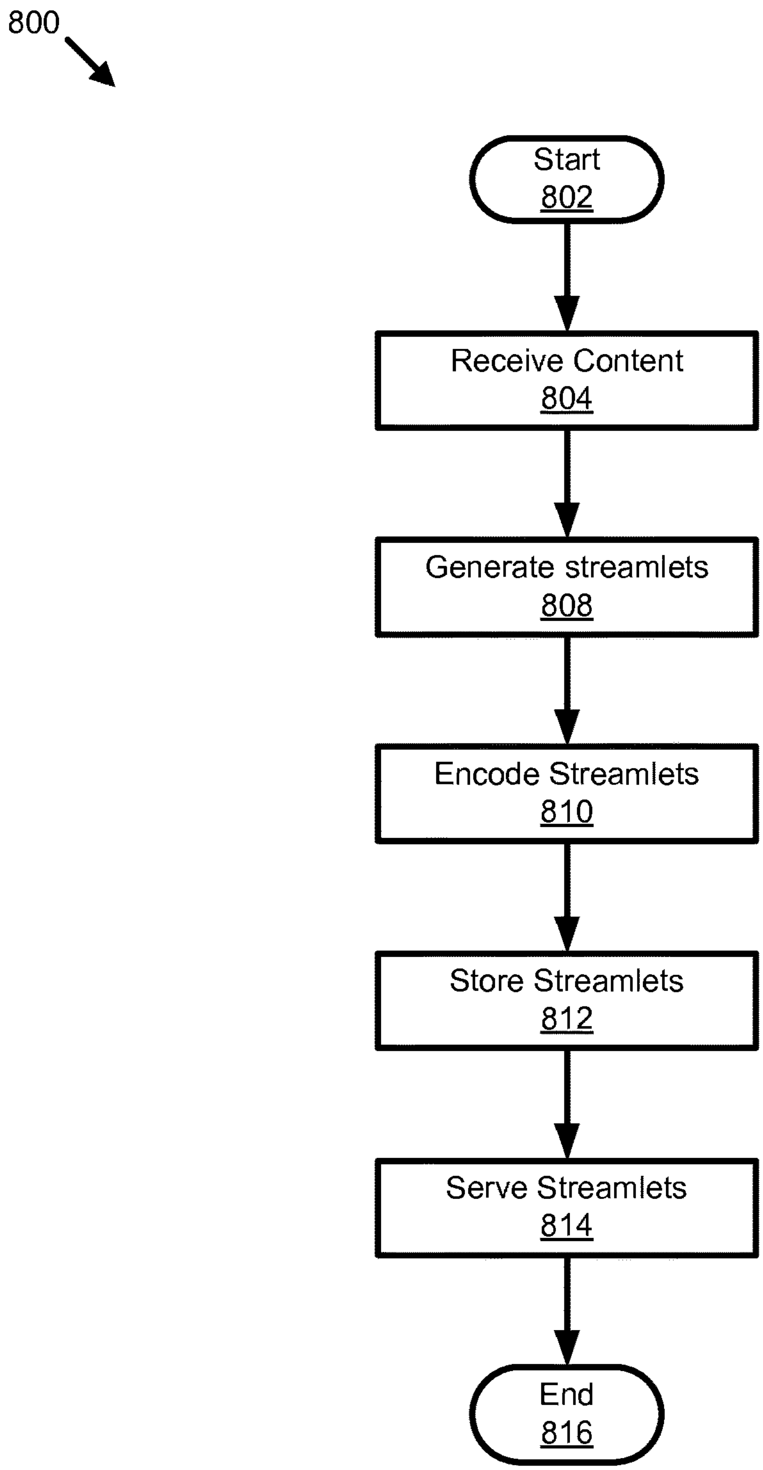
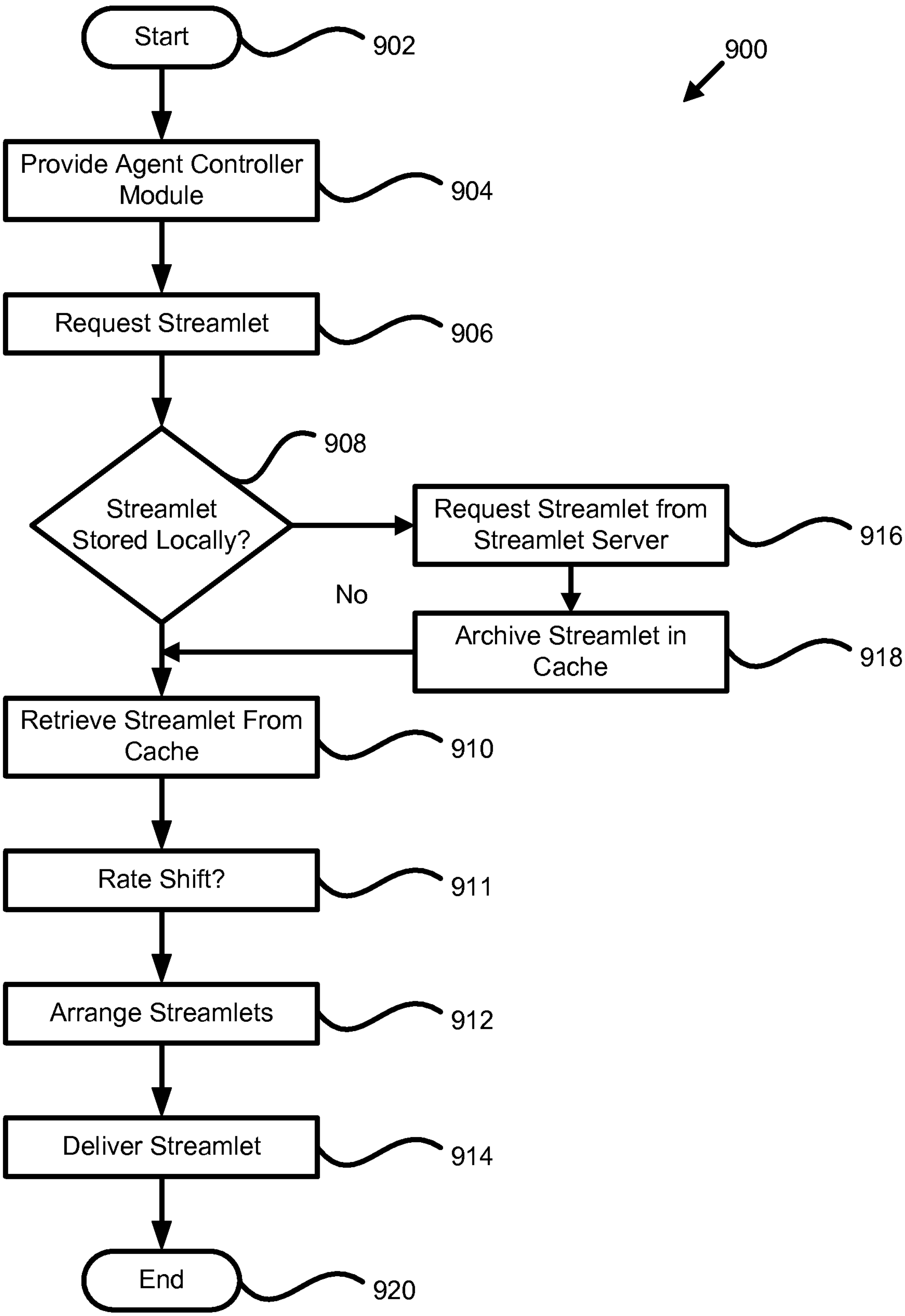
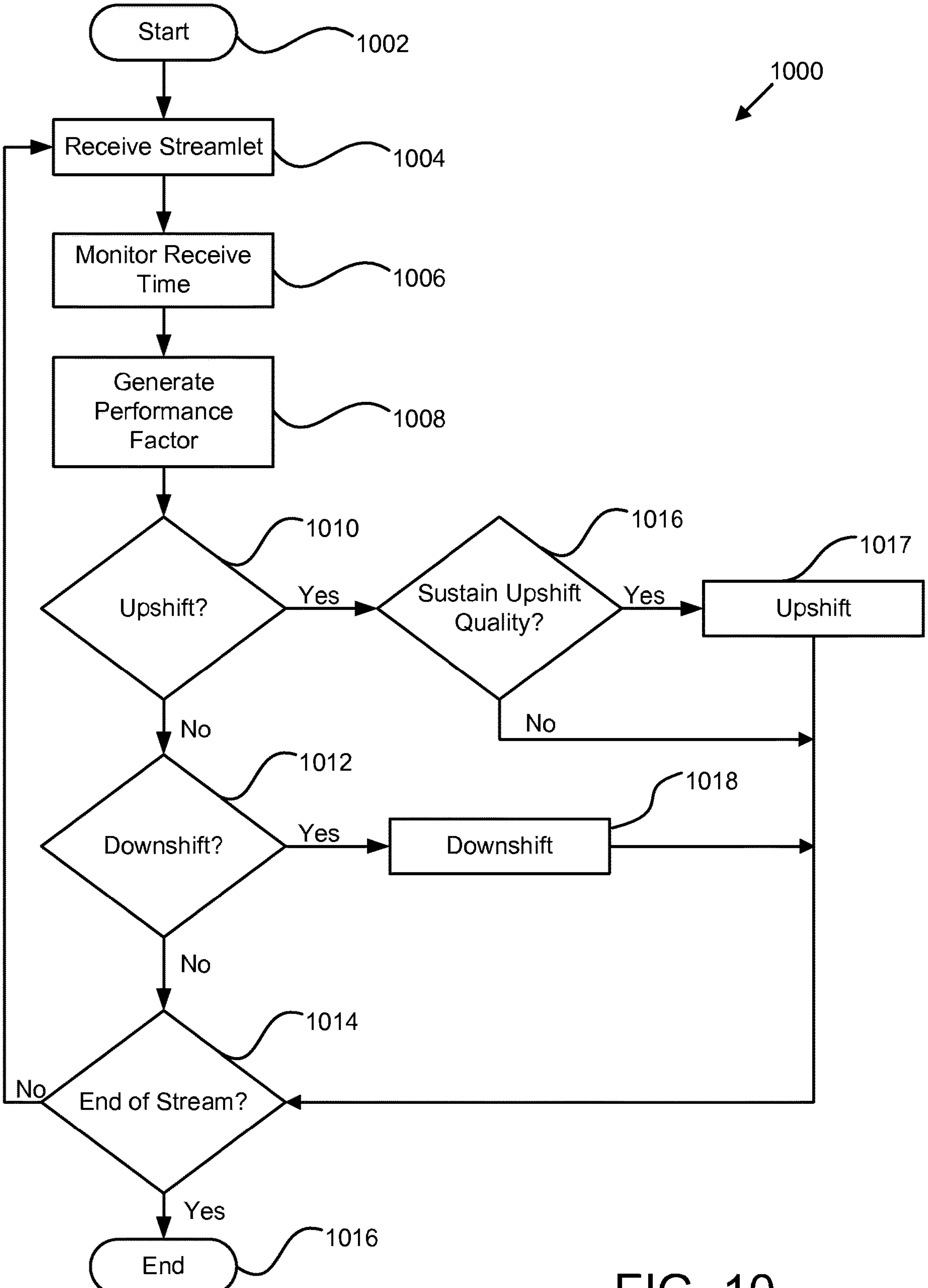

800
Start
802
Receive Content
804
Generate streamlets
808
Encode Streamlets
810
Store Streamlets
812
Serve Streamlets
814
End
816

FIG. 8



Start
902
900
Provide Agent Controller Module
904
Request Streamlet
906
908
Streamlet Stored Locally?
Request Streamlet from Streamlet Server
916
No
Archive Streamlet in Cache
918
Retrieve Streamlet From Cache
910
Rate Shift?
911
Arrange Streamlets
912
Deliver Streamlet
914
End
920

FIG. 9



Start
1002
1000
Receive Streamlet
1004
Monitor Receive Time
1006
Generate Performance Factor
1008
1010
Upshift?
Yes
1016
Sustain Upshift Quality?
Yes
1017
Upshift
No
No
1012
Downshift?
Yes
1018
Downshift
No
1014
No
End of Stream?
Yes
End
1016

FIG. 10

APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 16/004,056 filed on Jun. 8, 2018, which is a continuation of U.S. patent application Ser. No. 15/414,025 (now U.S. Pat. No. 9,998,516) filed on Jan. 24, 2017, which is a continuation of U.S. patent application Ser. No. 14/719,122 filed on May 21, 2015, which is a continuation of U.S. patent application Ser. No. 14/106,051 filed on Dec. 13, 2013 (now U.S. Pat. No. 9,071,668), which is a continuation of U.S. patent application Ser. No. 13/617,114, filed on Sep. 14, 2012 (now U.S. Pat. No. 8,612,624), which is a continuation of U.S. patent Ser. No. 12/906,940 filed on Oct. 18, 2010 (now U.S. Pat. No. 8,402,156), which is a continuation of U.S. patent application Ser. No. 11/673,483, filed on Feb. 9, 2007 (now U.S. Pat. No. 7,818,444), which is a continuation-in-part of application Ser. No. 11/116,783, filed on Apr. 28, 2005 (now U.S. Pat. No. 8,868,772), which claims the benefit of U.S. Provisional Application No. 60/566,831, filed on Apr. 31, 2004, all of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to video streaming over packet switched networks such as the Internet, and more particularly relates to adaptive-rate shifting of streaming content over such networks.

Description of the Related Art

The Internet is fast becoming a preferred method for distributing media files to end users. It is currently possible to download music or video to computers, cell phones, or practically any network capable device. Many portable media players are equipped with network connections and enabled to play music or videos. The music or video files (hereinafter "media files") can be stored locally on the media player or computer, or streamed or downloaded from a server.

"Streaming media" refers to technology that delivers content at a rate sufficient for presenting the media to a user in real time as the data is received. The data may be stored in memory temporarily until played and then subsequently deleted. The user has the immediate satisfaction of viewing the requested content without waiting for the media file to completely download. Unfortunately, the audio/video quality that can be received for real time presentation is constrained by the available bandwidth of the user's network connection. Streaming may be used to deliver content on demand (previously recorded) or from live broadcasts.

Alternatively, media files may be downloaded and stored on persistent storage devices, such as hard drives or optical storage, for later presentation. Downloading complete media files can take large amounts of time depending on the network connection. Once downloaded, however, the content can be viewed repeatedly anytime or anywhere. Media files prepared for downloading usually are encoded with a higher quality audio/video than can be delivered in real time. Users generally dislike this option, as they tend to want to see or hear the media file instantaneously.

Streaming offers the advantage of immediate access to the content but currently sacrifices quality compared with downloading a file of the same content. Streaming also provides the opportunity for a user to select different content for viewing on an ad hoc basis, while downloading is by definition restricted to receiving a specific content selection in its entirety or not at all. Downloading also supports rewind, fast forward, and direct seek operations, while streaming is unable to fully support these functions. Streaming is also vulnerable to network failures or congestion.

Another technology, known as "progressive downloads," attempts to combine the strengths of the above two technologies. When a progressive download is initiated, the media file download begins, and the media player waits to begin playback until there is enough of the file downloaded that playback can begin with the hope that the remainder of the file will be completely downloaded before playback "catches up." This waiting period before playback can be substantial depending on network conditions, and therefore is not a complete or fully acceptable solution to the problem of media presentation over a network.

Generally, three basic challenges exist with regard to data transport streaming over a network such as the Internet that has a varying amount of data loss. The first challenge is reliability. Most streaming solutions use a TCP connection, or "virtual circuit," for transmitting data. A TCP connection provides a guaranteed delivery mechanism so that data sent from one endpoint will be delivered to the destination, even if portions are lost and retransmitted. A break in the continuity of a TCP connection can have serious consequences when the data must be delivered in real-time. When a network adapter detects delays or losses in a TCP connection, the adapter "backs off" from transmission attempts for a moment and then slowly resumes the original transmission pace. This behavior is an attempt to alleviate the perceived congestion. Such a slowdown is detrimental to the viewing or listening experience of the user and therefore is not acceptable.

The second challenge to data transport is efficiency. Efficiency refers to how well the user's available bandwidth is used for delivery of the content stream. This measure is directly related to the reliability of the TCP connection. When the TCP connection is suffering reliability problems, a loss of bandwidth utilization results. The measure of efficiency sometimes varies suddenly, and can greatly impact the viewing experience.

The third challenge is latency. Latency is the time measure form the client's point-of-view, of the interval between when a request is issued and the response data begins to arrive. This value is affected by the network connection's reliability and efficiency, and the processing time required by the origin to prepare the response. A busy or overloaded server, for example, will take more time to process a request. As well as affecting the start time of a particular request, latency has a significant impact on the network throughput of TCP.

From the foregoing discussion, it should be apparent that a need exists for an apparatus, system, and method that alleviate the problems of reliability, efficiency, and latency. Additionally, such an apparatus, system, and method would offer instantaneous viewing along with the ability to fast forward, rewind, direct seek, and browse multiple streams. Beneficially, such an apparatus, system, and method would utilize multiple connections between a source and destination, requesting varying bitrate streams depending upon network conditions.

SUMMARY OF THE INVENTION

The present invention has been developed in response to the present state of the art, and in particular, in response to the problems and needs in the art that have not yet been fully solved by currently available content streaming systems. Accordingly, the present invention has been developed to provide an apparatus, system, and method for adaptive-rate content streaming that overcome many or all of the above-discussed shortcomings in the art.

The apparatus for adaptive-rate content streaming is provided with a logic unit containing a plurality of modules configured to functionally execute the necessary steps. These modules in the described embodiments include a receiving module configured to receive media content, a streamlet module configured to segment the media content and generate a plurality of sequential streamlets, and an encoding module configured to encode each streamlet as a separate content file.

The encoding module is further configured to generate a set of streamlets for each of the sequential streamlets. Each streamlet may comprise a portion of the media content having a predetermined length of time. The predetermined length of time may be in the range of between about 0.1 and 5 seconds.

In one embodiment, a set of streamlets comprises a plurality of streamlets having identical time indices, and each streamlet of the set of streamlets has a unique bitrate. The receiving module is configured to convert the media content to raw audio or raw video. The encoding module may include a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid. The job completion bid may be based on a plurality of computing variables selected from a group consisting of current encoding job completion percentage, average encoding job completion time, processor speed, and physical memory capacity.

A system of the present invention is also presented for adaptive-rate content streaming. In particular, the system, in one embodiment, includes a receiving module configured to receive media content, a streamlet module configured to segment the media content and generate a plurality of sequential streamlets, each streamlet comprising a portion of the media content having a predetermined length of time, and an encoding module configured to encode each streamlet as a separate content file and generate a set of streamlets.

The system also includes a plurality of streamlets having identical time indices and each streamlet of the set of streamlets having a unique bitrate. The encoding module comprises a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid.

A method of the present invention is also presented for adaptive-rate content streaming. In one embodiment, the method includes receiving media content, segmenting the media content and generating a plurality of sequential streamlets, and encoding each streamlet as a separate content file.

The method also includes segmenting the media content into a plurality of streamlets, each streamlet comprising a portion of the media content having a predetermined length of time. In one embodiment, the method includes generating a set of streamlets comprising a plurality of streamlets having identical time indices, and each streamlet of the set of streamlets having a unique bitrate.

Furthermore, the method may include converting the media content to raw audio or raw video, and segmenting the content media into a plurality of sequential streamlets. The method further comprises assigning an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid, and submitting an encoding job completion bid based on a plurality of computing variables.

Reference throughout this specification to features, advantages, or similar language does not imply that all of the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussion of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

Furthermore, the described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize that the invention may be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

These features and advantages of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the advantages of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings, in which:

FIG. **1** is a schematic block diagram illustrating one embodiment of a system for dynamic rate shifting of streaming content in accordance with the present invention;

FIG. **2*a*** is a schematic block diagram graphically illustrating one embodiment of a media content file;

FIG. **2*b*** is a schematic block diagram illustrating one embodiment of a plurality of streams having varying degrees of quality and bandwidth;

FIG. **3*a*** is a schematic block diagram illustrating one embodiment of a stream divided into a plurality of source streamlets;

FIG. **3*b*** is a schematic block diagram illustrating one embodiment of sets of streamlets in accordance with the present invention;

FIG. **4** is a schematic block diagram illustrating in greater detail one embodiment of the content module in accordance with the present invention;

FIG. **5*a*** is a schematic block diagram illustrating one embodiment of an encoder module in accordance with the present invention;

FIG. **5***b* is a schematic block diagram illustrating one embodiment of parallel encoding of streamlets in accordance with the present invention:

FIG. **6***a* is a schematic block diagram illustrating one embodiment of a virtual timeline in accordance with the present invention;

FIG. **6***b* is a schematic block diagram illustrating an alternative embodiment of a VT in accordance with the present invention:

FIG. **6***c* is a schematic block diagram illustrating one embodiment of a QMX in accordance with the present invention;

FIG. **7** is a schematic block diagram graphically illustrating one embodiment of a client module in accordance with the present invention:

FIG. **8** is a schematic flow chart diagram illustrating one embodiment of a method for processing content in accordance with the present invention;

FIG. **9** is a schematic flow chart diagram illustrating one embodiment of a method for viewing a plurality of streamlets in accordance with the present invention; and

FIG. **10** is a schematic flow chart diagram illustrating one embodiment of a method for requesting streamlets within an adaptive-rate shifting content streaming environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Many of the functional units described in this specification have been labeled as modules, in order to more particularly emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom VLSI circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices or the like.

Modules may also be implemented in software for execution by various types of processors. An identified module of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions which may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may comprise disparate instructions stored in different locations which, when joined logically together, comprise the module and achieve the stated purpose for the module.

Indeed, a module of executable code may be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

Reference throughout this specification to "one embodiment," "an embodiment." or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in one embodiment," "in an embodiment," and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

Reference to a signal bearing medium may take any form capable of generating a signal, causing a signal to be generated, or causing execution of a program of machine-readable instructions on a digital processing apparatus. A signal bearing medium may be embodied by a transmission line, a compact disk, digital-video disk, a magnetic tape, a Bernoulli drive, a magnetic disk, a punch card, flash memory, integrated circuits, or other digital processing apparatus memory device. In one embodiment, a computer program product including a computer useable medium having a computer readable program of computer instructions stored thereon that when executed on a computer causes the computer to carry out operations for multi-bitrate content streaming as described herein.

Furthermore, the described features, structures, or characteristics of the invention may be combined in any suitable manner in one or more embodiments. In the following description, numerous specific details are provided, such as examples of programming, software modules, user selections, network transactions, database queries, database structures, hardware modules, hardware circuits, hardware chips, etc., to provide a thorough understanding of embodiments of the invention. One skilled in the relevant art will recognize, however, that the invention may be practiced without one or more of the specific details, or with other methods, components, materials, and so forth. In other instances, well-known structures, materials, or operations are not shown or described in detail to avoid obscuring aspects of the invention.

FIG. **1** is a schematic block diagram illustrating one embodiment of a system **100** for dynamic rate shifting of streaming content in accordance with the present invention. In one embodiment, the system **100** comprises a content server **102** and an end user station **104**. The content server **102** and the end user station **104** may be coupled by a data communications network. The data communications network may include the Internet **106** and connections **108** to the Internet **106**. Alternatively, the content server **102** and the end user **104** may be located on a common local area network, wireless area network, cellular network, virtual local area network, or the like. The end user station **104** may comprise a personal computer (PC), an entertainment system configured to communicate over a network, or a portable electronic device configured to present content. For example, portable electronic devices may include, but are not limited to, cellular phones, portable gaming systems, and portable computing devices.

In the depicted embodiment, the system **100** also includes a publisher **110**, and a web server **116**. The publisher **110** may be a creator or distributor of content. For example, if the content to be streamed were a broadcast of a television program, the publisher **110** may be a television or cable network channel such as NBC®, or MTV®. Content may be transferred over the Internet **106** to the content server **102**, where the content is received by a content module **112**. The content module **112** may be configured to receive, process, and store content. In one embodiment, processed content is accessed by a client module **114** configured to play the content on the end user station **104**. In a further embodiment, the client module **114** is configured to receive different portions of a content stream from a plurality of locations simultaneously. For example, the client module **114** may request and receive content from any of the plurality of web servers **116**.

Content from the content server **102** may be replicated to other web servers **116** or alternatively to proxy cache servers **118**. Replicating may occur by deliberate forwarding from the content server **102**, or by a web, cache, or proxy server outside of the content server **102** asking for content on behalf of the client module **114**. In a further embodiment, content may be forwarded directly to web **116** or proxy **118** servers through direct communication channels **120** without the need to traverse the Internet **106**.

FIG. **2***a* is a schematic block diagram graphically illustrating one embodiment of a media content (hereinafter "content") file **200**. In one embodiment, the content file **200** is distributed by the publisher **110**. The content file **200** may comprise a television broadcast, sports event, movie, music, concert, etc. The content file **200** may also be live or archived content. The content file **200** may comprise uncompressed video and audio, or alternatively, video or audio. Alternatively, the content file **200** may be compressed using standard or proprietary encoding schemes. Examples of encoding schemes capable of use with the present invention include, but are not limited to, DivX®, Windows Media Video®, Quicktime Sorenson 3®, On2, OGG Vorbis, MP3, or Quicktime 6.5/MPEG-4® encoded content.

FIG. **2***b* is a schematic block diagram illustrating one embodiment of a plurality of streams **202** having varying degrees of quality and bandwidth. In one embodiment, the plurality of streams **202** comprises a low quality stream **204**, a medium quality stream **206**, and a high quality stream **208**. Each of the streams **204**, **206**, **208** is a copy of the content file **200** encoded and compressed to varying bit rates. For example, the low quality stream **204** may be encoded and compressed to a bit rate of 100 kilobits per second (kbps), the medium quality stream **206** may be encoded and compressed to a bit rate of 200 kbps, and the high quality stream **208** may be encoded and compressed to 600 kbps.

FIG. **3***a* is a schematic block diagram illustrating one embodiment of a stream **302** divided into a plurality of source streamlets **303**. As used herein, streamlet refers to any sized portion of the content file **200**. Each streamlet **303** may comprise a portion of the content contained in stream **302**, encapsulated as an independent media object. The content in a streamlet **303** may have a unique time index in relation to the beginning of the content contained in stream **302**. In one embodiment, the content contained in each streamlet **303** may have a duration of two seconds. For example, streamlet **0** may have a time index of 00:00 representing the beginning of content playback, and streamlet **1** may have a time index of 00:02, and so on. Alternatively, the time duration of the streamlets **304** may be any duration smaller than the entire playback duration of the content in stream **302**. In a further embodiment, the streamlets **303** may be divided according to file size instead of a time index and duration.

FIG. **3***b* is a schematic block diagram illustrating one embodiment of sets **306** of streamlets in accordance with the present invention. As used herein, the term "set" refers to a group of streamlets having identical time indices and durations but varying bitrates. In the depicted embodiment, the set **306***a* encompasses all streamlets having a time index of 00:00. The set **306***a* includes encoded streamlets **304** having low, medium, and high 204, 206, 208 bitrates. Of course each set **306** may include more than the depicted three bitrates which are given by way of example only. One skilled in the art will recognize that any number of streams having different bitrates may be generated from the original content **200**.

As described above, the duration of one streamlet **304** may be approximately two seconds. Likewise each set **306** may comprise a plurality of streamlets **304** where each streamlet **304** has a playable duration of two seconds. Alternatively, the duration of the streamlet **304** may be predetermined or dynamically variable depending upon a variety of factors including, but not limited to, network congestion, system specifications, playback resolution and quality, etc. In the depicted embodiment, the content **200** may be formed of the plurality of sets **306**. The number of sets **306** may depend on the length of the content **200** and the length or duration of each streamlet **304**.

FIG. **4** is a schematic block diagram illustrating in greater detail one embodiment of the content module **112** in accordance with the present invention. The content module **112** may comprise a capture module **402**, a streamlet module **404**, an encoder module **406**, a streamlet database **408**, and the web server **116**. In one embodiment, the capture module **402** is configured to receive the content file **200** from the publisher **110**. The capture module **402** may be configured to "decompress" the content file **200**. For example, if the content file **200** arrives having been encoded with one of the above described encoding schemes, the capture module **402** may convert the content file **200** into raw audio and/or video. Alternatively, the content file **200** may be transmitted by the publisher in a format **110** that does not require decompression.

The capture module **402** may comprise a capture card configured for TV and/or video capture. One example of a capture card suitable for use in the present invention is the DRC-2500 by Digital Rapids of Ontario, Canada. Alternatively, any capture card capable of capturing audio and video may be utilized with the present invention. In a further embodiment, the capture module **402** is configured to pass the content file to the streamlet module **404**.

The streamlet module **404**, in one embodiment, is configured to segment the content file **200** and generate source streamlets **303** that are not encoded. As used herein, the term "segment" refers to an operation to generate a streamlet of the content file **200** having a duration or size equal to or less than the duration or size of the content file **200**. The streamlet module **404** may be configured to segment the content file **200** into streamlets **303** each having an equal duration. Alternatively, the streamlet module **404** may be configured to segment the content file **200** into streamlets **303** having equal file sizes.

The encoding module **406** is configured to receive the source streamlets **303** and generate the plurality of streams **202** of varying qualities. The original content file **200** from the publisher may be digital in form and may comprise content having a high bit rate such as, for example, 2 mbps. The content may be transferred from the publisher **110** to the content module **112** over the Internet **106**. Such transfers of data are well known in the art and do not require further discussion herein. Alternatively, the content may comprise a captured broadcast.

In a further embodiment, the encoding module **406** is configured to generate a plurality of sets **306** of streamlets **304**. The sets **306**, as described above with reference to FIG. **3***b*, may comprise streamlets having an identical time index and duration, and a unique bitrate. As with FIG. **3***b*, the sets **306** and subsequently the plurality of streams **202** may comprise the low quality stream **204**, the medium quality stream **206**, and the high quality stream **208**. Alternatively, the plurality of streams **202** may comprise any number of streams deemed necessary to accommodate end user bandwidth.

The encoder module **406** is further configured to encode each source streamlet **303** into the plurality of streams **202** and streamlet sets **306** and store the streamlets in the streamlet database **408**. The encoding module **406** may utilize encoding schemes such as DivX®, Windows Media Video 9®, Quicktime 6.5 Sorenson 3®, or Quicktime 6.5/MPEG-4®. Alternatively, a custom encoding scheme may be employed.

The content module **112** may also include a metadata module **412** and a metadata database **414**. In one embodiment, metadata comprises static searchable content information. For example, metadata includes, but is not limited to, air date of the content, title, actresses, actors, length, and episode name. Metadata is generated by the publisher **110**, and may be configured to define an end user environment. In one embodiment, the publisher **100** may define an end user navigational environment for the content including menus, thumbnails, sidebars, advertising, etc. Additionally, the publisher **110** may define functions such as fast forward, rewind, pause, and play that may be used with the content file **200**. The metadata module **412** is configured to receive the metadata from the publisher **110** and store the metadata in the metadata database **414**. In a further embodiment, the metadata module **412** is configured to interface with the client module **114**, allowing the client module **114** to search for content based upon at least one of a plurality of metadata criteria. Additionally, metadata may be generated by the content module **112** through automated process(es) or manual definition.

Once the streamlets **304** have been received and processed, the client module **114** may request streamlets **304** using HTTP from the web server **116**. Using a standard protocol such as HTTP eliminates the need for network administrators to configure firewalls to recognize and pass through network traffic for a new, specialized protocol. Additionally, since the client module **114** initiates the request, the web server **116** is only required to retrieve and serve the requested streamlet **304**. In a further embodiment, the client module **114** may be configured to retrieve streamlets **304** from a plurality of web servers **116**.

Each web server **116** may be located in various locations across the Internet **106**. The streamlets **304** may essentially be static files. As such, no specialized media server or server-side intelligence is required for a client module **114** to retrieve streamlets **304**. Streamlets **304** may be served by the web server **116** or cached by cache servers of Internet Service Providers (ISPs), or any other network infrastructure operators, and served by the cache server. Use of cache servers is well known to those skilled in the art, and will not be discussed further herein. Thus, a highly scalable solution is provided that is not hindered by massive amounts of client module **114** requests to the web server **116** at any specific location, especially the web server **116** most closely associated with or within the content module **112**

FIG. **5***a* is a schematic block diagram illustrating one embodiment of an encoder module **406** in accordance with the present invention. In one embodiment, the encoder module **406** may include a master module **502** and a plurality of host computing modules (hereinafter "host") **504**. The hosts **504** may comprise personal computers, servers, etc. In a further embodiment, the hosts **504** may be dedicated hardware, for example, cards plugged into a single computer.

The master module (hereinafter "master") **502** is configured to receive streamlets **303** from the streamlet module **404** and stage the streamlet **303** for processing. In one embodiment, the master **502** may decompress each source streamlet **303** to produce a raw streamlet. As used herein, the term "raw streamlet" refers to a streamlet **303** that is uncompressed or lightly compressed to substantially reduce size with no significant loss in quality. A lightly compressed raw streamlet can be transmitted more quickly and to more hosts. Each host **504** is coupled with the master **502** and configured to receive a raw streamlet from the master **502** for encoding. The hosts **504**, in one example, generate a plurality of streamlets **304** having identical time indices and durations, and varying bitrates. Essentially each host **504** may be configured to generate a set **306** from the raw streamlet **503** sent from the master **502**. Alternatively, each host **504** may be dedicated to producing a single bitrate in order to reduce the time required for encoding.

Upon encoding completion, the host **504** returns the set **306** to the master **502** so that the encoding module **406** may store the set **306** in the streamlet database **408**. The master **502** is further configured to assign encoding jobs to the hosts **504**. Each host is configured to submit an encoding job completion bid (hereinafter "bid"). The master **502** assigns encoding jobs depending on the bids from the hosts **504**. Each host **504** generates a bid depending upon a plurality of computing variables which may include, but are not limited to, current encoding job completion percentage, average job completion time, processor speed and physical memory capacity.

For example, a host **504** may submit a bid that indicates that based on past performance history the host **504** would be able to complete the encoding job in 15 seconds. The master **502** is configured to select from among a plurality of bids the best bid and subsequently submit the encoding job to the host **504** with the best bid. As such, the described encoding system does not require that each host **504** have identical hardware but beneficially takes advantage of the available computing power of the hosts **504**. Alternatively, the master **502** selects the host **504** based on a first come first serve basis, or some other algorithm deemed suitable for a particular encoding job.

The time required to encode one streamlet **304** is dependent upon the computing power of the host **504**, and the encoding requirements of the content file **200**. Examples of encoding requirements may include, but are not limited to, two or multi-pass encoding, and multiple streams of different bitrates. One benefit of the present invention is the ability to perform two-pass encoding on a live content file **200**. Typically, in order to perform two-pass encoding prior art systems must wait for the content file to be completed before encoding.

The present invention, however, segments the content file **200** into source streamlets **303** and the two-pass encoding to a plurality of streams **202** may be performed on each corresponding raw streamlet without waiting for a TV show to end, for example. As such, the content module **112** is capable of streaming the streamlets over the Internet shortly after the content module **112** begins capture of the content file **200**. The delay between a live broadcast transmitted from the publisher **110** and the availability of the content depends on the computing power of the hosts **504**.

FIG. **5***b* is a schematic block diagram illustrating one embodiment of parallel encoding of streamlets in accordance with the present invention. In one example, the capture module **402** (of FIG. **4**) begins to capture the content file and the streamlet module **404** generates a first streamlet **303***a* and passes the streamlet to the encoding module **406**. The encoding module **406** may take 10 seconds, for example, to generate the first set **306***a* of streamlets **304***a* (**304***a***1**, **304***a***2**, **304***a***3**, etc. represent streamlets **304** of

different bitrates). FIG. **5***b* illustrates the encoding process generically as block **502** to graphically illustrate the time duration required to process a raw or lightly encoded streamlet **303** as described above with reference to the encoding module **406**. The encoding module **406** may simultaneously process more than one streamlet **303**, and processing of streamlets will begin upon arrival of the streamlet from the capture module **402**.

During the 10 seconds required to encode the first streamlet **303***a*, the streamlet module **404** has generated five additional 2-second streamlets **303***b*, **303***c*, **303***d*, **303***e*, **303***f*, for encoding and the master **502** has prepared and staged the corresponding raw streamlets. Two seconds after the first set **306***a* is available the next set **306***b* is available, and so on. As such, the content file **200** is encoded for streaming over the Internet and appears live. The 10 second delay is given herein by way of example only. Multiple hosts **504** may be added to the encoding module **406** in order to increase the processing capacity of the encoding module **406**. The delay may be shortened to an almost unperceivable level by the addition of high CPU powered systems, or alternatively multiple low powered systems.

A system as described above beneficially enables multi-pass encoding of live events. Multi-pass encoding systems of the prior art require that the entire content be captured (or be complete) because in order to perform multi-pass encoding the entire content must be scanned and processed more than once. This is impossible with prior art systems because content from a live event is not complete until the event is over. As such, with prior art systems, multi-pass encoding can only be performed once the event is over. Streamlets, however, may be encoded as many times as is deemed necessary. Because the streamlet is an encapsulated media object of 2 seconds (for example), multi-pass encoding may begin on a live event once the first streamlet is captured. Shortly after multi-pass encoding of the first streamlet **303***a* is finished, multi-pass encoding of the second streamlet **303***b* finishes, and as such multi-pass encoding is performed on a live event and appears live to a viewer.

Any specific encoding scheme applied to a streamlet may take longer to complete than the time duration of the streamlet itself, for example, a very high quality encoding of a 2-second streamlet may take 5 seconds to finish. Alternatively, the processing time required for each streamlet may be less than the time duration of a streamlet. However, because the offset parallel encoding of successive streamlets are encoded by the encoding module at regular intervals (matching the intervals at which the those streamlets are submitted to the encoding module **406**, for example 2 seconds) the output timing of the encoding module **406** does not fall behind the real-time submission rate of the unencoded streamlets. Conversely, prior art encoding systems rely on the very fastest computing hardware and software because the systems must generate the output immediately in lock-step with the input. A prior art system that takes 2.1 seconds to encode 2 seconds worth of content is considered a failure. The present invention allows for slower than real-time encoding processes yet still achieves a real-time encoding effect due to the parallel offset pipes.

The parallel offset pipeline approach described with reference to FIG. **5***b* beneficially allows for long or short encoding times without "falling behind" the live event. Additionally, arbitrarily complex encoding of streamlets to multiple profiles and optimizations only lengthens the encoding time **502** without a perceptible difference to a user because the sets **306** of streamlets **304** are encoded in a time-selective manner so that streamlets are processed at regular time intervals and transmitted at these time intervals.

Returning now to FIG. **5***a*, as depicted, the master **502** and the hosts **504** may be located within a single local area network, or in other terms, the hosts **504** may be in close physical proximity to the master **502**. Alternatively, the hosts **504** may receive encoding jobs from the master **502** over the Internet or other communications network. For example, consider a live sports event in a remote location where it would be difficult to setup multiple hosts. In this example, a master performs no encoding or alternatively light encoding before publishing the streamlets online. The hosts **504** would then retrieve those streamlets and encode the streamlets into the multiple bitrate sets **306** as described above.

Furthermore, hosts **504** may be dynamically added or removed from the encoding module without restarting the encoding job and/or interrupting the publishing of streamlets. If a host **504** experiences a crash or some failure, its encoding work is simply reassigned to another host.

The encoding module **406**, in one embodiment, may also be configured to produce streamlets that are specific to a particular playback platform. For example, for a single raw streamlet, a single host **504** may produce streamlets for different quality levels for personal computer playback, streamlets for playback on cell phones with a different, proprietary codec, a small video-only streamlet for use when playing just a thumbnail view of the stream (like in a programming guide), and a very high quality streamlet for use in archiving.

FIG. **6***a* is a schematic block diagram illustrating one embodiment of a virtual timeline **600** in accordance with the present invention. In one embodiment, the virtual timeline **600** comprises at least one quantum media extension **602**. The quantum media extension (hereinafter "QMX") **602** describes an entire content file **200**. Therefore, the virtual timeline (hereinafter "VT") **600** may comprise a file that is configured to define a playlist for a user to view. For example, the VT may indicate that the publisher desires a user to watch a first show QMX **602***a* followed by QMX **602***b* and QMX **602***c*. As such, the publisher may define a broadcast schedule in a manner similar to a television station.

FIG. **6***b* is a schematic block diagram illustrating an alternative embodiment of a VT **600** in accordance with the present invention. In the depicted embodiment, the VT **600** may include a single QMX **602** which indicates that the publisher desires the same content to be looped over and over again. For example, the publisher may wish to broadcast a never-ending infomercial on a website.

FIG. **6***c* is a schematic block diagram illustrating one embodiment of a QMX **602** in accordance with the present invention. In one embodiment, the QMX **602** contains a multitude of information generated by the content module **112** configured to describe the content file **200**. Examples of information include, but are not limited to, start index **604**, end index **606**, whether the content is live **608**, proprietary publisher data **610**, encryption level **612**, content duration **614** and bitrate values **616**. The bitrate values **616** may include frame size **618**, audio channel **620** information, codecs **622** used, sample rate **624**, and frames parser **626**.

A publisher may utilize the QVT **600** together with the QMX **602** in order to prescribe a playback order for users, or alternatively selectively edit content. For example, a publisher may indicate in the QMX **602** that audio should be muted at time index 10:42 or video should be skipped for 3 seconds at time index 18:35. As such, the publisher may

selectively skip offensive content without the processing requirements of editing the content.

FIG. **7** is a schematic block diagram graphically illustrating one embodiment of a client module **114** in accordance with the present invention. The client module **114** may comprise an agent controller module **702**, a streamlet cache module **704**, and a network controller module **706**. In one embodiment, the agent controller module **702** is configured to interface with a viewer **708**, and transmit streamlets **304** to the viewer **708**. Alternatively, the agent controller module **702** may be configured to simply reassemble streamlets into a single file for transfer to an external device such as a portable video player.

In a further embodiment, the client module **114** may comprise a plurality of agent controller modules **702**. Each agent controller module **702** may be configured to interface with one viewer **708**. Alternatively, the agent controller module **702** may be configured to interface with a plurality of viewers **708**. The viewer **708** may be a media player (not shown) operating on a PC or handheld electronic device.

The agent controller module **702** is configured to select a quality level of streamlets to transmit to the viewer **708**. The agent controller module **702** requests lower or higher quality streams based upon continuous observation of time intervals between successive receive times of each requested streamlet. The method of requesting higher or lower quality streams will be discussed in greater detail below with reference to FIG. **10**.

The agent controller module **702** may be configured to receive user commands from the viewer **708**. Such commands may include play, fast forward, rewind, pause, and stop. In one embodiment, the agent controller module **702** requests streamlets **304** from the streamlet cache module **704** and arranges the received streamlets **304** in a staging module **709**. The staging module **709** may be configured to arrange the streamlets **304** in order of ascending playback time. In the depicted embodiment, the streamlets **304** are numbered 0, 1, 2, 3, 4, etc. However, each streamlet **304** may be identified with a unique filename.

Additionally, the agent controller module **702** may be configured to anticipate streamlet **304** requests and pre-request streamlets **304**. By pre-requesting streamlets **304**, the user may fast-forward, skip randomly, or rewind through the content and experience no buffering delay. In a further embodiment, the agent controller module **702** may request the streamlets **304** that correspond to time index intervals of 30 seconds within the total play time of the content. Alternatively, the agent controller module **702** may request streamlets at any interval less than the length of the time index. This enables a "fast-start" capability with no buffering wait when starting or fast-forwarding through content file **200**. In a further embodiment, the agent controller module **702** may be configured to pre-request streamlets **304** corresponding to specified index points within the content or within other content in anticipation of the end user **104** selecting new content to view. In one embodiment, the streamlet cache module **704** is configured to receive streamlet **304** requests from the agent controller module **702**. Upon receiving a request, the streamlet cache module **704** first checks a streamlet cache **710** to verify if the streamlet **304** is present. In a further embodiment, the streamlet cache module **704** handles streamlet **304** requests from a plurality of agent controller modules **702**. Alternatively, a streamlet cache module **704** may be provided for each agent controller module **702**. If the requested streamlet **304** is not present in the streamlet cache **410**, the request is passed to the network controller module **706**. In order to enable fast forward and rewind capabilities, the streamlet cache module **704** is configured to store the plurality of streamlets **304** in the streamlet cache **710** for a specified time period after the streamlet **304** has been viewed. However, once the streamlets **304** have been deleted, they may be requested again from the web server **116**.

The network controller module **706** may be configured to receive streamlet requests from the streamlet cache module **704** and open a connection to the web server **116** or other remote streamlet **304** database (not shown). In one embodiment, the network controller module **706** opens a TCP/IP connection to the web server **116** and generates a standard HTTP GET request for the requested streamlet **304**. Upon receiving the requested streamlet **304**, the network controller module **706** passes the streamlet **304** to the streamlet cache module **704** where it is stored in the streamlet cache **710**. In a further embodiment, the network controller module **706** is configured to process and request a plurality of streamlets **304** simultaneously. The network controller module **706** may also be configured to request a plurality of streamlets, where each streamlet **304** is subsequently requested in multiple parts.

In a further embodiment, streamlet requests may comprise requesting pieces of any streamlet file. Splitting the streamlet **304** into smaller pieces or portions beneficially allows for an increased efficiency potential, and also eliminates problems associated with multiple full-streamlet requests sharing the bandwidth at any given moment. This is achieved by using parallel TCP/IP connections for pieces of the streamlets **304**. Consequently, efficiency and network loss problems are overcome, and the streamlets arrive with more useful and predictable timing.

In one embodiment, the client module **114** is configured to use multiple TCP connections between the client module **114** and the web server **116** or web cache. The intervention of a cache may be transparent to the client or configured by the client as a forward cache. By requesting more than one streamlet **304** at a time in a manner referred to as "parallel retrieval," or more than one part of a streamlet **304** at a time, efficiency is raised significantly and latency is virtually eliminated. In a further embodiment, the client module allows a maximum of three outstanding streamlet **304** requests. The client module **114** may maintain additional open TCP connections as spares to be available should another connection fail. Streamlet **304** requests are rotated among all open connections to keep the TCP flow logic for any particular connection from falling into a slow-start or close mode. If the network controller module **706** has requested a streamlet **304** in multiple parts, with each part requested on mutually independent TCP/IP connections, the network controller module **706** reassembles the parts to present a complete streamlet **304** for use by all other components of the client module **114**.

When a TCP connection fails completely, a new request may be sent on a different connection for the same streamlet **304**. In a further embodiment, if a request is not being satisfied in a timely manner, a redundant request may be sent on a different connection for the same streamlet **304**. If the first streamlet request's response arrives before the redundant request response, the redundant request can be aborted. If the redundant request response arrives before the first request response, the first request may be aborted.

Several streamlet **304** requests may be sent on a single TCP connection, and the responses are caused to flow back in matching order along the same connection. This eliminates all but the first request latency. Because multiple responses are always being transmitted, the processing

latency of each new streamlet **304** response after the first is not a factor in performance. This technique is known in the industry as "pipelining." Pipelining offers efficiency in request-response processing by eliminating most of the effects of request latency. However, pipelining has serious vulnerabilities. Transmission delays affect all of the responses. If the single TCP connection fails, all of the outstanding requests and responses are lost. Pipelining causes a serial dependency between the requests.

Multiple TCP connections may be opened between the client module **114** and the web server **116** to achieve the latency-reduction efficiency benefits of pipelining while maintaining the independence of each streamlet **304** request. Several streamlet **304** requests may be sent concurrently, with each request being sent on a mutually distinct TCP connection. This technique is labeled "virtual pipelining" and is an innovation of the present invention. Multiple responses may be in transit concurrently, assuring that communication bandwidth between the client module **114** and the web server **116** is always being utilized. Virtual pipelining eliminates the vulnerabilities of traditional pipelining. A delay in or complete failure of one response does not affect the transmission of other responses because each response occupies an independent TCP connection. Any transmission bandwidth not in use by one of multiple responses (whether due to delays or TCP connection failure) may be utilized by other outstanding responses.

A single streamlet **304** request may be issued for an entire streamlet **304**, or multiple requests may be issued, each for a different part or portion of the streamlet. If the streamlet is requested in several parts, the parts may be recombined by the client module **114** streamlet.

In order to maintain a proper balance between maximized bandwidth utilization and response time, the issuance of new streamlet requests must be timed such that the web server **116** does not transmit the response before the client module **114** has fully received a response to one of the previously outstanding streamlet requests. For example, if three streamlet **304** requests are outstanding, the client module **114** should issue the next request slightly before one of the three responses is fully received and "out of the pipe." In other words, request timing is adjusted to keep three responses in transit. Sharing of bandwidth among four responses diminishes the net response time of the other three responses. The timing adjustment may be calculated dynamically by observation, and the request timing adjusted accordingly to maintain the proper balance of efficiency and response times.

The schematic flow chart diagrams that follow are generally set forth as logical flow chart diagrams. As such, the depicted order and labeled steps are indicative of one embodiment of the presented method. Other steps and methods may be conceived that are equivalent in function, logic, or effect to one or more steps, or portions thereof, of the illustrated method. Additionally, the format and symbols employed are provided to explain the logical steps of the method and are understood not to limit the scope of the method. Although various arrow types and line types may be employed in the flow chart diagrams, they are understood not to limit the scope of the corresponding method. Indeed, some arrows or other connectors may be used to indicate only the logical flow of the method. For instance, an arrow may indicate a waiting or monitoring period of unspecified duration between enumerated steps of the depicted method. Additionally, the order in which a particular method occurs may or may not strictly adhere to the order of the corresponding steps shown.

FIG. **8** is a schematic flow chart diagram illustrating one embodiment of a method **800** for processing content in accordance with the present invention. In one embodiment the method **800** starts **802**, and the content module **112** receives **804** content from the publisher **110**. Receiving content **804** may comprise receiving **804** a digital copy of the content file **200**, or digitizing a physical copy of the content file **200**. Alternatively, receiving **804** content may comprise capturing a radio, television, cable, or satellite broadcast. Once received **804**, the streamlet module **404** generates **808** a plurality of source streamlets **303** each having a fixed duration. Alternatively, the streamlets **303** may be generated with a fixed file size.

In one embodiment, generating **808** streamlets comprises dividing the content file **200** into a plurality of two second streamlets **303**. Alternatively, the streamlets may have any length less than or equal to the length of the stream **202**. The encoder module **406** then encodes **810** the streamlets **303** into sets **306** of streamlets **304**, in a plurality of streams **202** according to an encoding scheme. The quality may be predefined, or automatically set according to end user bandwidth, or in response to pre-designated publisher guidelines

In a further embodiment, the encoding scheme comprises a proprietary codec such as WMV9®. The encoder module **406** then stores **812** the encoded streamlets **304** in the streamlet database **408**. Once stored **812**, the web server **116** may then serve **814** the streamlets **304**. In one embodiment, serving **814** the streamlets **304** comprises receiving streamlet requests from the client module **114**, retrieving the requested streamlet **304** from the streamlet database **408**, and subsequently transmitting the streamlet **304** to the client module **114**. The method **800** then ends **816**.

FIG. **9** is a schematic flow chart diagram illustrating one embodiment of a method **900** for viewing a plurality of streamlets in accordance with the present invention. The method **900** starts and an agent controller module **702** is provided **904** and associated with a viewer **708** and provided with a staging module **709**. The agent controller module **702** then requests **906** a streamlet **304** from the streamlet cache module **704**. Alternatively, the agent controller module **702** may simultaneously request **906** a plurality of streamlets **304** the streamlet cache module **704**. If the streamlet is stored **908** locally in the streamlet cache **710**, the streamlet cache module **704** retrieves **910** the streamlet **304** and sends the streamlet to the agent controller module **702**. Upon retrieving **910** or receiving a streamlet, the agent controller module **702** makes **911** a determination of whether or not to shift to a higher or lower quality stream **202**. This determination will be described below in greater detail with reference to FIG. **10**.

In one embodiment, the staging module **709** then arranges **912** the streamlets **304** into the proper order, and the agent controller module **702** delivers **914** the streamlets to the viewer **708**. In a further embodiment, delivering **914** streamlets **304** to the end user comprises playing video and or audio streamlets on the viewer **708**. If the streamlets **304** are not stored **908** locally, the streamlet request is passed to the network controller module **706**. The network controller module **706** then requests **916** the streamlet **304** from the web server **116**. Once the streamlet **304** is received, the network controller module **706** passes the streamlet to the streamlet cache module **704**. The streamlet cache module **704** archives **918** the streamlet. Alternatively, the streamlet cache module **704** then archives **918** the streamlet and passes the streamlet to the agent controller module **702**, and the method **900** then continues from operation **910** as described above.

Referring now to FIG. **10**, shown therein is a schematic flow chart diagram illustrating one embodiment of a method **1000** for requesting streamlets **304** within an adaptive-rate shifting content streaming environment in accordance with the present invention. The method **1000** may be used in one embodiment as the operation **911** of FIG. **9**. The method **1000** starts and the agent controller module **702** receives **1004** a streamlet **304** as described above with reference to FIG. **9**. The agent controller module **702** then monitors **1006** the receive time of the requested streamlet. In one embodiment, the agent controller module **702** monitors the time intervals A between successive receive times for each streamlet response. Ordering of the responses in relation to the order of their corresponding requests is not relevant.

Because network behavioral characteristics fluctuate, sometimes quite suddenly, any given Δ may vary substantially from another. In order to compensate for this fluctuation, the agent controller module **702** calculates **1008** a performance ratio r across a window of n samples for streamlets of playback length S. In one embodiment, the performance ratio r is calculated using the equation:

$$r = S\frac{n}{\sum_{i-1}^{n}\Delta_i}$$

Due to multiple simultaneous streamlet processing, and in order to better judge the central tendency of the performance ratio r, the agent controller module **702** may calculate a geometric mean, or alternatively an equivalent averaging algorithm, across a window of size m, and obtain a performance factor φ:

$$\varphi_{current} = \left(\prod_{j=1}^{m} r_j\right)^{\frac{1}{m}}$$

The policy determination about whether or not to upshift **1010** playback quality begins by comparing $\varphi_{current}$ with a trigger threshold Θ_{up}. If $\varphi_{current} \geq \Theta_{up}$, then an up shift to the next higher quality stream may be considered **1016**. In one embodiment, the trigger threshold Θ_{up} is determined by a combination of factors relating to the current read ahead margin (i.e. the amount of contiguously available streamlets that have been sequentially arranged by the staging module **709** for presentation at the current playback time index), and a minimum safety margin. In one embodiment, the minimum safety margin may be 24 seconds. The smaller the read ahead margin, the larger Θ_{up} is to discourage upshifting until a larger read ahead margin may be established to withstand network disruptions. If the agent controller module **702** is able to sustain **1016** upshift quality, then the agent controller module **702** will upshift **1017** the quality and subsequently request higher quality streams. The determination of whether use of the higher quality stream is sustainable **1016** is made by comparing an estimate of the higher quality stream's performance factor, φ_{higher}, with Θ_{up}. If $\varphi_{higher} \geq \Theta_{up}$ then use of the higher quality stream is considered sustainable. If the decision of whether or not the higher stream rate is sustainable **1016** is "no," the agent controller module **702** will not attempt to upshift **1017** stream quality. If the end of the stream has been reached **1014**, the method **1000** ends **1016**.

If the decision on whether or not to attempt upshift **1010** is "no", a decision about whether or not to downshift **1012** is made. In one embodiment, a trigger threshold Θ_{down} is defined in a manner analogous to Θ_{up}. If $\varphi_{current} > \Theta_{down}$ then the stream quality may be adequate, and the agent controller module **702** does not downshift **1018** stream quality. However, if $\varphi_{current} \leq \Theta_{down}$, the agent controller module **702** does downshift **1018** the stream quality. If the end of the stream has not been reached **1014**, the agent controller module **702** begins to request and receive **1004** lower quality streamlets and the method **1000** starts again. Of course, the above described equations and algorithms are illustrative only, and may be replaced by alternative streamlet monitoring solutions.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. A system for adaptive-rate content streaming of video playable on one or more end user stations over the Internet, the system comprising:

at least one processor executing non-transitory executable instructions for generating at least one virtual timeline corresponding to the video;

wherein the video encoded at a plurality of different bitrates creating a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, the low quality stream, the medium quality stream, and the high quality stream each comprising a group of streamlets encoded at a respective one of the plurality of different bitrates, each group of streamlets comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and

wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the video in each of the low quality stream, the medium quality stream, and the high quality stream, and wherein the first streamlet of the low quality stream encodes the same first portion of the video at a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream.

2. The system of claim **1**, wherein the processor is further for generating a plurality of virtual timelines wherein each virtual timeline corresponds to each of t the low quality stream, the medium quality stream, and the high quality stream.

3. The system of claim **1**, wherein the video is a live event video.

4. The system of claim **1**, wherein the video includes archived content.

5. The system of claim **1**, wherein the second streamlet of each of the groups of streamlets each has the same second duration and corresponds to the same second portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the second streamlet of the low quality stream having the same bitrate as the first streamlet of the low quality stream.

6. The system of claim 5, wherein the first and second durations are different.

7. The system of claim 1, further comprising: a

plurality of web servers located at different locations across the internet, each web server configured to: receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing a portion of the video, wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams; retrieve from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and send the retrieved first streamlet from the currently selected one of the different copies to the requesting one of the end user stations over the one or more network connections.

8. The system of claim 1, further comprising:

a first web server configured to:

receive at least one virtual timeline request over the one or more internet connections from the one or more end user stations to retrieve a virtual timeline; and send the virtual timeline to the requesting one of the end user stations over the one or more network connections.

9. The system of claim 8, wherein the first web server is further configured to:

receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the video,

wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the video; retrieve from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and send the retrieved first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream to the requesting one of the end user stations over the one or more network connections.

10. The system of claim 1, wherein the at least one virtual timeline corresponds to the currently selected one of the low quality stream, the medium quality stream, and the high quality stream.

11. The system of claim 1, wherein the virtual timeline defines a playlist for a user to view.

12. The system of claim 1, wherein the virtual timeline comprises a file that is configured to define a playlist for a user to view.

13. The system of claim 12, wherein the virtual timeline comprises at least one quantum media extension (QMX).

14. An end user station to stream a video over a network from a server for playback of the video, the content player device comprising:

a processor;

a digital processing apparatus memory device comprising non-transitory machine-readable instructions that, when executed, cause the processor to:

establish one or more network connections between the end user station and the server, wherein the server is configured to access at least one of a plurality of groups of streamlets;

wherein the video is encoded at a plurality of different bitrates to create a plurality of streams including at least a low quality stream, a medium quality stream, and a high quality stream, each of the low quality stream, the medium quality stream, and the high quality stream comprising a group of streamlets encoded at the same respective one of the different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bit rate of no less than 600 kbps; and wherein the first streamlets of each of the low quality stream, the medium quality stream and the high quality stream each has an equal playback duration and each of the first streamlets encodes the same portion of the video at a different one of the different bitrates;

select a specific one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams;

place at least one virtual timeline request for at least one virtual times based on the selected one of the he low quality stream, the medium quality stream, and the high quality stream; and

receive the at least one virtual timeline.

15. The end user station of claim 14, wherein the non-transitory machine-readable instructions that, when executed, further cause the processor to:

place one or more streamlet requests to the server over the one or more network connections for the first streamlet of the selected stream; receive the requested first streamlet from the server via the one or more network connections wherein the one or more streamlet requests are based on the at least one virtual timeline; and

provide the received first streamlet for playback of the video.

16. The end user station of claim 14, wherein the second streamlet of each of the groups of streamlets each has the same second duration and corresponds to the same second portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the second streamlet of the low quality stream having the same bitrate as the first streamlet of the low quality stream.

17. The end user station of claim 16, wherein the first and second durations are different.

18. The end user station of claim 17, wherein the virtual timeline corresponds to the currently selected one of the low quality stream, the medium quality stream, and the high quality stream.

19. The end user station of claim 18, wherein the virtual timeline defines a playlist for a user to view.

20. The end user station of claim 14, wherein the video is a live event video.

21. The end user station of claim 14, wherein the video includes archived content.

22. A process executable by one or more servers to stream a video for playback by one or more end user stations, the process comprising:

storing, by the one or more servers, one or more virtual timelines corresponding to a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the first streamlet of the low quality stream having a different one of the different bitrates than the first streamlet of the high quality stream and the first streamlet of the medium quality stream;

receiving at least one virtual timeline request over one or more internet connections from the one or more end user stations to retrieve a virtual timeline correspond to the first streamlet storing the first portion of the video,

wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the video;

retrieving from the storage device the requested virtual timeline for the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

sending the retrieved virtual timeline to the requesting one of the end user stations over the one or more network connections.

23. The process of claim **22**, further comprising:

storing, by the one or more servers, a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream; and

receiving at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the video,

wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the video

retrieving from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

sending the retrieved first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream to the requesting one of the end user stations over the one or more network connections.

24. The process of claim **22**, wherein the second streamlet of each of the groups of streamlets each has the same second duration and corresponds to the same second portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the second streamlet of the low quality stream having the same bitrate as the first streamlet of the low quality stream.

25. The process of claim **22**, wherein the first and second durations are different.

26. The process of claim **22**, wherein the video is a live event video.

27. The process of claim **22**, wherein the video includes archived content.

28. A process executable by a content player device to stream a video over a network from a server for playback of the video by the content player device, the process comprising:

establishing one or more network connections between the content player device and the server,

wherein the server accesses a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video; wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and

wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the first streamlet of the low quality stream having a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream;

selecting, by the content player device, a currently selected one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the video;

placing a virtual time request over one or more internet connections from the one or more end user stations to retrieve at least one virtual timeline corresponding to the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

receiving the requested virtual timeline from the server via the one or more network connections.

29. The process of claim **28** further comprising: placing a streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the video, wherein the streamlet request is based, at least in part, on the received virtual timeline;

receiving the requested streamlet from the server via the one or more network connections; and rendering, by the content player device, the received streamlet for playback of the video.

* * * * *

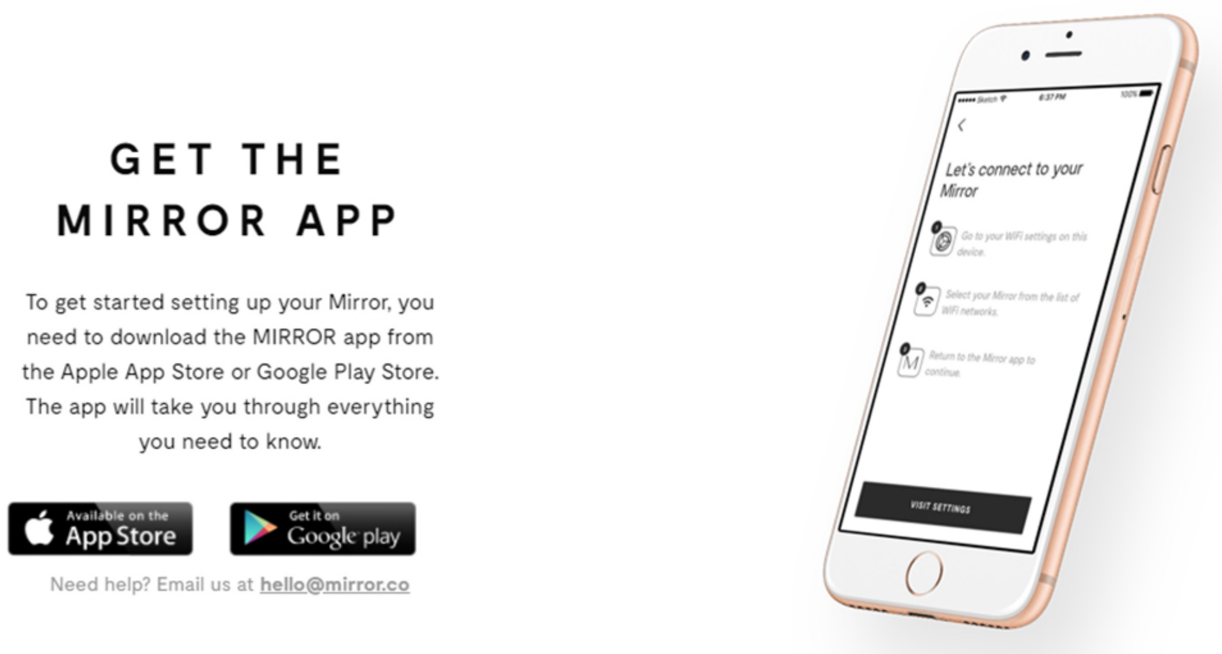
EXHIBIT E-1

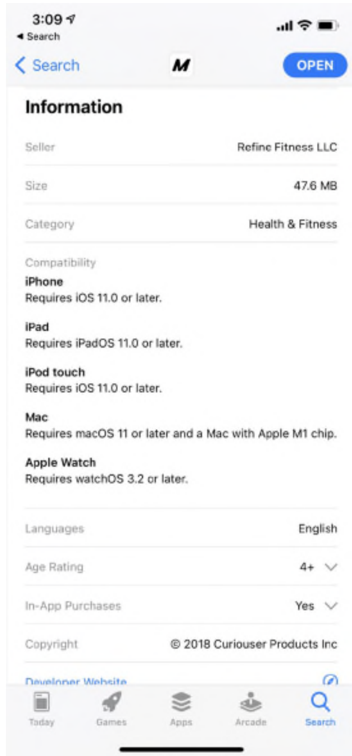
U.S. Patent No. 10,951,680 to Mirror

The following claim chart shows exemplary aspects of the Mirror Application and Mirror Device that infringe the claims below. The chart is exemplary and should not be read to limit DISH's claims against Mirror to the specific products or services described below. The chart should also not be read to limit DISH's claims to the patent claims charted below. Nor should the chart below be read to limit how the Mirror Application and Mirror Devices infringe the claims below.

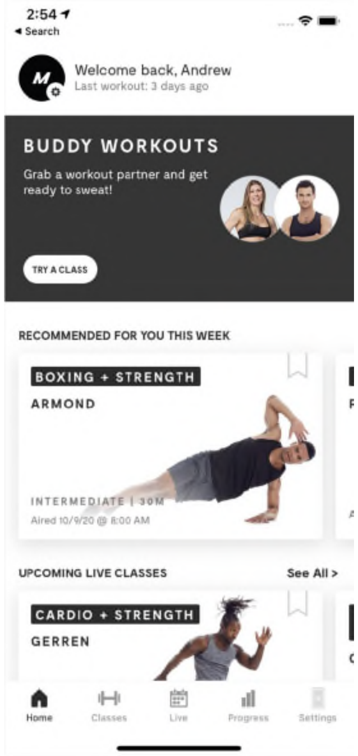
Claim	Claim Limitation	Example Infringement Evidence
14	An end user station to stream a video over a network from a server for playback of the video, the content player device comprising:	The Mirror Application is software that permits "an end user station to stream a video over a network from a server for playback of the video." The Mirror Application is executable by devices that are end user stations and it obtains streams of a selected video for playback of the video. The streams are obtained by the Mirror Application over a network. The exemplary images in this chart of the Mirror Application are from the Mirror Application running on an Apple iPhone XS (Mirror's iOS Application). In addition, the Mirror Application is available to run on other devices. Unless otherwise noted, each of these devices is an "end user station" with a content player device. **MIRROR DIGITAL OVERVIEW** Access all of MIRROR's Live and On Demand Classes 24/7 from your phone, tablet, and smart TV devices. This feature is included for free with the MIRROR Membership and access is available immediately (whether you already have your Mirror or are awaiting delivery). Please note, it is not currently possible to purchase MIRROR Digital if you do not own a Mirror. https://mirror.kustomer.help/en_us/mirror-digital-overview-B1pnVd8UU

USP 10,951,680 to Mirror

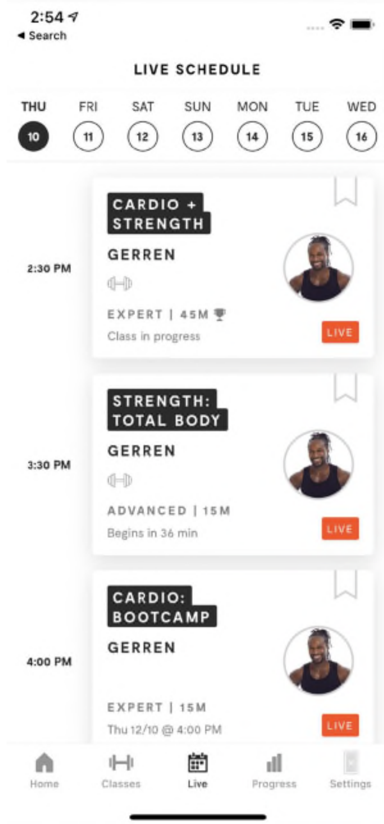
Claim	Claim Limitation	Example Infringement Evidence
		 https://www.mirror.co/app. **MIRROR APP** The MIRROR App allows you to access and customize the Mirror experience. The MIRROR App is available for both iOS and Android! • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-app--S1dDC_tYm.

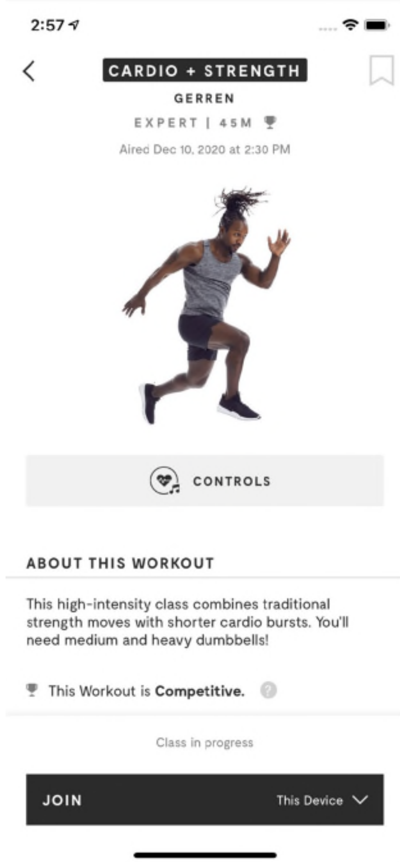
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Apple App Store When launched, the Mirror Application displays a main menu:

USP 10,951,680 to Mirror

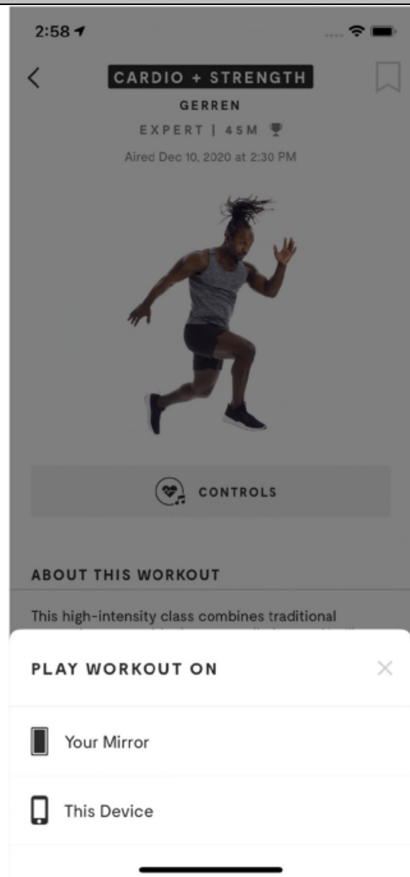
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application The main menu of the Mirror Application displays classes that are each a "video." The "Upcoming Live Classes" section of the Mirror Application main menu displays a preview of ongoing and upcoming live video programming.

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting a video from the list causes the Mirror Application to display more details regarding the class and provides the option to join the class.

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting the "Join" button for a class causes the Mirror iOS Application to provide options to stream the class to a variety of end user stations over the Internet, including the iOS device that the user is using or the separate Mirror Device.

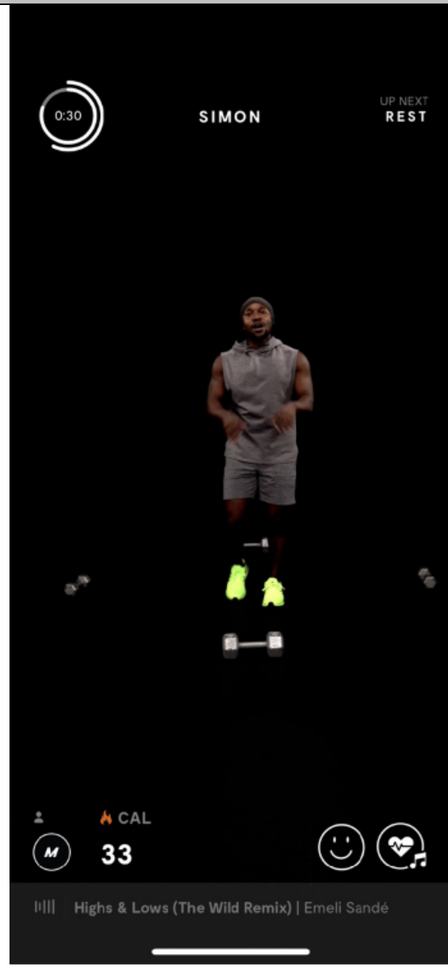
USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting "Your Mirror" causes the selected video and other material to be streamed on the user's Mirror device, which is connected to the Internet.

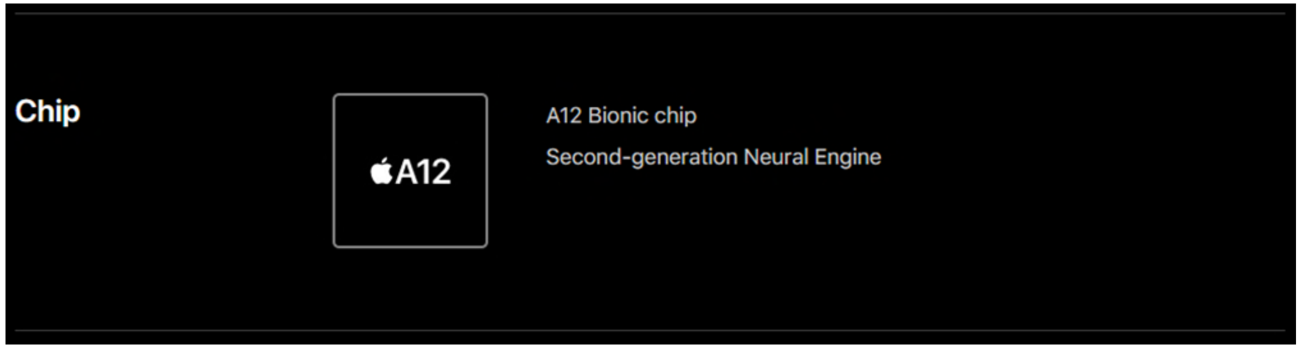
USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Alternatively, selecting "This Device" causes the selected workout video and other material to be streamed on the user's iOS device:

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application As shown above, the Mirror Devices are "end user station[s] to stream a video over a network from a server for playback of the video." The Mirror Devices obtain streams of a selected video program for playback of the video on the Mirror Device's content player, as shown above. The streams are obtained over a network.

Claim	Claim Limitation	Example Infringement Evidence
	a processor;	A device running the Mirror Application includes at least one "processor" to execute the Mirror Application and stream the live event video. The devices that are compatible with the Mirror Application, each include one or more processors. **MIRROR DIGITAL COMPATIBLE DEVICES** The MIRROR App is available for the iPhone, iPad, Android phones, and Android tablets. MIRROR Digital can be casted to your smart TV using these devices. • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-digital-compatible-devices-HklDdOU8U. For example, Mirror requires users to provide a user device such as an iPhone that includes a processor to execute the Mirror Application. **Information** Seller: Refine Fitness LLC Size: 99.3 MB Category: Health & Fitness Compatibility: Requires iOS 11.0 and watchOS 3.2 or later. Compatible with iPhone, iPad, and iPod touch. Languages: English Age Rating: 4+ Copyright: © 2018 Curiouser Products Inc Price: Free In-App Purchases: 1. 1 Year Subscription for Mirror $599.99 Developer Website ↗ App Support ↗ Privacy Policy ↗ https://apps.apple.com/us/app/mirror-workout-companion/id1153358600.

Claim	Claim Limitation	Example Infringement Evidence
		 https://www.apple.com/iphone-xr/specs/. The Mirror Devices also include a processor.

Claim	Claim Limitation	Example Infringement Evidence
		HARDWARE FRAME: Carbon steel frame; Mineral bronze powder coated DISPLAY: 40" full HD 1080p display, with 178° wide viewing angle TECHNOLOGY: Quad core processor SOUND: 2 x 10 watt high-fidelity stereo speakers; Embedded omnidirectional microphone CAMERA: 5 megapixel front-facing camera POWER: 1 ft and 6 ft right angle UL certified cables https://www.mirror.co/shop/mirror.
	a digital processing apparatus memory device comprising non-transitory machine-readable instructions that,	The device executes the Mirror Application from "a digital processing apparatus memory device comprising non-transitory machine-readable instructions." The instructions include at least the executable instructions for the Mirror Application and its features. Mirror requires users of the Mirror Application to provide a device with a digital processing apparatus memory device to store the instructions.

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
	when executed, cause the processor to:	For example, Mirror requires users to provide at least 99.3 MB of storage on a digital processing apparatus memory device of the end user station for storing the Mirror Application. **Information** Seller: Refine Fitness LLC Size: 99.3 MB Category: Health & Fitness Compatibility: Requires iOS 11.0 and watchOS 3.2 or later. Compatible with iPhone, iPad, and iPod touch. Languages: English Age Rating: 4+ Copyright: © 2018 Curiouser Products Inc Price: Free In-App Purchases: 1. 1 Year Subscription for Mirror $599.99 Developer Website ↗ App Support ↗ Privacy Policy ↗ https://apps.apple.com/us/app/mirror-workout-companion/id1153358600. The Mirror Devices also include "a digital processing apparatus memory device comprising non-transitory machine-readable instructions." For example, the on-board quad core processor requires memory containing non-transitory machine-readable instructions in order to process and display streaming fitness classes.

Claim	Claim Limitation	Example Infringement Evidence
		HARDWARE FRAME — Carbon steel frame Mineral bronze powder coated DISPLAY — 40" full HD 1080p display, with 178° wide viewing angle TECHNOLOGY — Quad core processor SOUND — 2 x 10 watt high-fidelity stereo speakers Embedded omnidirectional microphone CAMERA — 5 megapixel front-facing camera POWER — 1 ft and 6 ft right angle UL certified cables https://www.mirror.co/shop/mirror.
	establish one or more network connections between the end user station and the server, wherein the server is configured to access	As shown below, the non-transitory machine-readable instructions of the Mirror Application and Mirror Devices, when executed, cause the processor(s) to "establish one or more network connections between the end user station and the server" that is "configured to access at least one of a plurality of groups of streamlets." The "segments" discussed herein are "streamlets." The Mirror Application requires an internet connection.

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
	at least one of a plurality of groups of streamlets;	**PRELOAD CLASSES ON MIRROR DIGITAL** You currently cannot preload classes on the MIRROR Digital, however this feature is coming soon! If you are not able to connect WiFi or are in a tough WiFi environment, you can always use cellular data to stream classes on the MIRROR App. Please consult your cell phone provider for questions about data usage and your plan. https://mirror.kustomer.help/en_us/preload-classes-on-mirror-app-H12XPdUUL. To stream a video, such as that shown above, the Mirror Application requests a stream of a selected video via a network connection. A user may select to stream a video by selecting the Join button, as shown above. When the Mirror Application accesses a selected video, it requests and receives a playlist file that shows the available versions of the program at different resolutions. For the following test, a live event video was selected. In the test, an iPhone 11 running the Mirror Application makes an HTTPS GET request for a master playlist named "**playlist.m3u8**" that specifies the available streams and provides links to the playlists for those streams. The following master playlist named "**playlist.m3u8**" is returned.

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
<pre>
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080
../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720
../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480
../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360
../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288
../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180
../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8
</pre>
Filename: playlist.m3u8.
This is a master playlist file according to the HLS specification.[1] The playlist shows six versions of the stream, denoted by each #EXT-X-STREAM-INF tag at the following bandwidths:

6434112 (referred to herein as “6434112 Bandwidth”)
864048 (referred to herein as “864048 Bandwidth”)
403824 (referred to herein as “403824 Bandwidth”)
367728 (referred to herein as “367728 Bandwidth”)
312832 (referred to herein as “312832 Bandwidth”)
249664 (referred to herein as “249664 Bandwidth”)

For each of these versions, the master playlist provides a link to a playlist file for the specified version of the selected live event video at a particular bandwidth and resolution, which is called a “variant” in HLS.
</td></tr>
</table>

[1] RFC 8216 (HLS Live Streaming), Section 4.3.4 (Master Playlist Tags)

Claim	Claim Limitation	Example Infringement Evidence
		The Mirror Application initially selects the **6434112 Bandwidth** (1080p – high bandwidth) version of the stream and makes a request for the corresponding variant playlist file named "**chunklist.m3u8**." That file with the following contents (a portion of which is shown below) is returned. #EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1232 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:08.356Z #EXTINF:2.0, r4vhrugx/00000000/media_1232.ts #EXTINF:2.0, r4vhrugx/00000000/media_1233.ts #EXTINF:2.0, r4vhrugx/00000000/media_1234.ts #EXTINF:2.0, r4vhrugx/00000000/media_1235.ts #EXTINF:2.0, r4vhrugx/00000000/media_1236.ts #EXTINF:2.0, r4vhrugx/00000000/media_1237.ts #EXTINF:2.0, r4vhrugx/00000000/media_1238.ts #EXTINF:2.0, r4vhrugx/00000000/media_1239.ts #EXTINF:2.0, r4vhrugx/00000000/media_1240.ts #EXTINF:2.0, r4vhrugx/00000000/media_1241.ts #EXTINF:2.0, Path: https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8

Claim	Claim Limitation	Example Infringement Evidence
		As noted above, the variant playlist file is an HLS playlist. The variant playlist file identifies a plurality of segments or “streamlets” that are part of the **6434112 Bandwidth** group of streamlets. Each line in the file “**chunklist.m3u8**” that begins with “#EXTINF” specifies the length of the segments in seconds (2.0). The line below the #EXTINF entry is the relative location of the video file for the segment (e.g., r4vhrugx/0000000/media_1232.ts). The Mirror Application uses HTTPS GET requests to retrieve the segments of the encoded live event video specified in the file above to access the segments. The Mirror Application makes a request for a segment to access and return the requested segment. The Mirror Application content player plays back the segment to stream the selected live event. As long as the viewer stays on the selected live event video and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **6434112 Bandwidth** version). While the example above relates to the **6434112 Bandwidth** group of streamlets, other groups of streamlets are available. If the available bandwidth for the network connection decreases, for example, the Mirror Application will continue to request segments to continue streaming the live event program, but at one of the lower bandwidths. For example, for the current test, the Mirror Application subsequently made a request for the variant playlist file for the **403824 Bandwidth** named “chunklist.m3u8.” The file with the following contents is returned showing a portion of the **403824 Bandwidth** group of segments for the live event video being streamed.

USP 10,951,680 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1238
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z
#EXTINF:2.0,
fbd862nq/00000000/media_1238.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1239.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1240.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1241.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1242.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1243.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1244.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1245.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1246.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1247.ts
#EXTINF:2.0,
fbd862nq/00000000/media_1248.ts

Filename: chunklist.m3u8

The Mirror Application then makes the request for fbd862nq/00000000/media_1238.ts. The requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected live event video at the 403824 Bandwidth. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror</td></tr>
</table>

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **403824 Bandwidth** version). As the bandwidth is further constrained, the Mirror Application makes another request for a lower quality stream. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variable playlist file for the **249664 Bandwidth** named "chunklist.m3u8." The file with the following contents is returned showing a portion of the **249664 Bandwidth** group of segments for the live event video being streamed. #EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1241 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:26.354Z #EXTINF:2.0, zf4q4ivl/00000000/media_1241.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1242.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1243.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1244.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1245.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1246.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1247.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1248.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1249.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1250.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1251.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1252.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1253.ts File: **chunklist.m3u8**

Claim	Claim Limitation	Example Infringement Evidence
		The Mirror Application then makes the request for zf4q4ivl/00000000/media_1281.ts. The requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected live event video at the **249664 Bandwidth**. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **249664 Bandwidth** version). A portion of a subsequently retrieved playlist's contents are shown below.

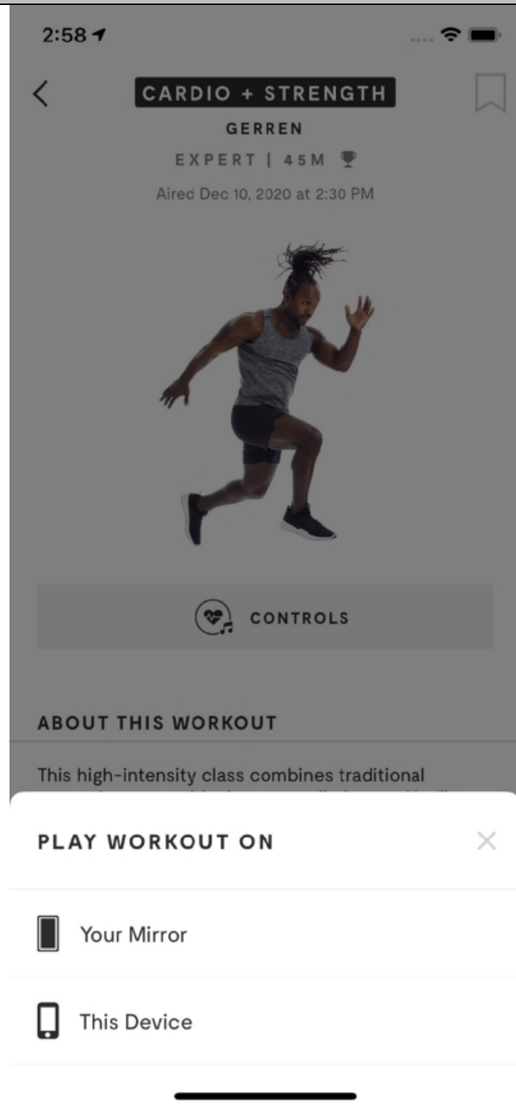
USP 10,951,680 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
<pre>
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1245
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:34.354Z
#EXTINF:2.0,
zf4q4ivl/00000000/media_1245.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1246.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1247.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1248.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1249.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1250.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1251.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1252.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1253.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1254.ts
#EXTINF:2.0,
zf4q4ivl/00000000/media_1255.ts
#EXTINF:2.0,
</pre>
Filename: chunklist.m3u8
The subsequently retrieved playlist includes additional video segments that were not included in the previous playlist file, for example: “media_1254.ts” and “media_1255.ts.” The Mirror Application continues to request, receive, and playback successive segments of the live event video to stream the live event video.
</td></tr>
</table>

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		The Mirror Devices also require an internet connection. **CONNECTION** INTERNET — Dual-band 802.11 A/B/G/N Wi-Fi APP — Controlled by iOS or Android companion app HEART RATE — Syncs with Bluetooth™ heart rate monitors, Apple Watches, and Android Wear OS Watches AUDIO — Pairs with Bluetooth™ speakers and headphones https://www.mirror.co/shop/mirror To stream a video, such as that shown above, the Mirror Device requests a stream of a selected video via a network connection. The iOS application provides the interface for interacting with a Mirror Device (i.e., selecting a live or on demand class to stream). After selecting a class, the user selects whether to use the iOS device or Mirror Device to view the content (i.e., stream and participate in the workout).

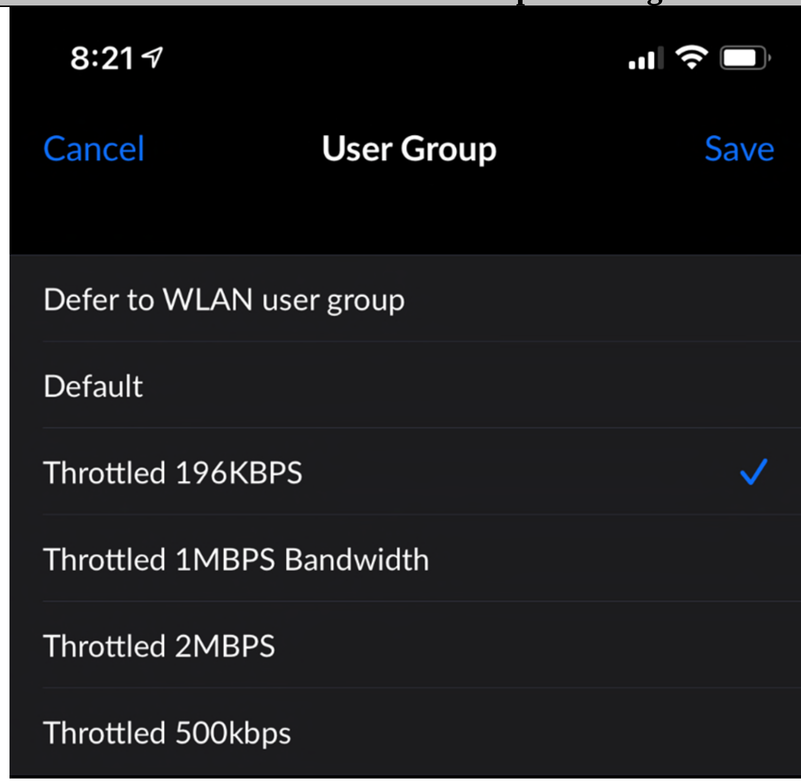
USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application

Claim	Claim Limitation	Example Infringement Evidence
		Selecting “Your Mirror” causes the Mirror Device to initiate streaming:  For the following test, a live programming event was selected. Based on the test, and upon information and belief, the Mirror Devices operate in substantially the same way as the Mirror Application. For example, when the Mirror Device(s) accesses a selected live event video, the Mirror Device(s) initially selects a first bandwidth version of the stream, makes a request for the segments of the group corresponding to the selected bandwidth version of the live event program, receives segments from the group corresponding to the selected bandwidth version, and then plays the requested and received segments on the Mirror Device content player as shown below.

USP 10,951,680 to Mirror

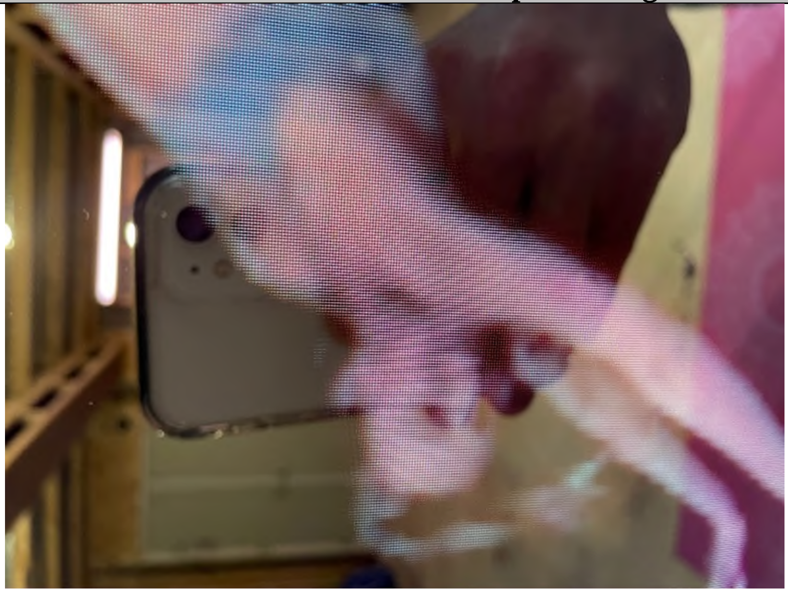
Claim	Claim Limitation	Example Infringement Evidence
		 Other groups of streamlets are available. For example, for the current test, bandwidth for the Mirror Device was constrained to 1Mbps, which caused the Mirror Device to display a "buffering" message while requesting and receiving a corresponding playlist and streamlets for a second bandwidth version of the live event video as shown below.

Claim	Claim Limitation	Example Infringement Evidence
		

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 The image resolution for the second bandwidth streamlet requested is noticeably lower quality as indicated by the pixelated edges of the instructor's body, as shown below.

USP 10,951,680 to Mirror

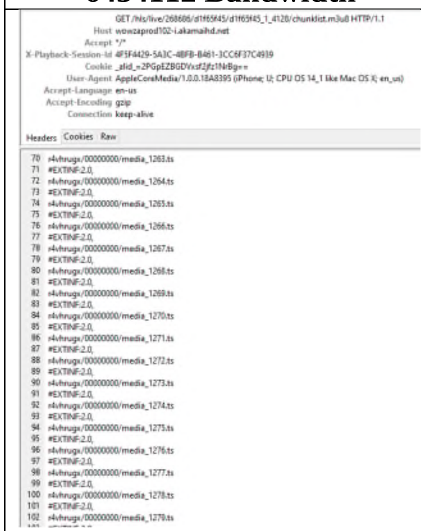
Claim	Claim Limitation	Example Infringement Evidence
		
	wherein the video is encoded at a plurality of different bitrates to create a plurality of streams including at least a low quality stream, a medium quality stream, and a high quality stream, each of the low quality stream, the medium quality stream, and the high quality stream comprising a group	The "video is encoded at a plurality of different bitrates to create a plurality of streams." The plurality of different bitrates creates a plurality of streams "including at least a low quality stream, a medium quality stream, and a high quality stream." Further, "each of the low quality stream, the medium quality stream, and the high quality stream compris[es] a group of streamlets encoded at the same respective one of the different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the live event video." As shown in the master playlist file, "**playlist.m3u8**," the video for the live event video is encoded at 6 different bitrates.

Claim	Claim Limitation	Example Infringement Evidence
	of streamlets encoded at the same respective one of the different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video;	#EXTM3U #EXT-X-VERSION:3 #EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080 ../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720 ../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480 ../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360 ../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288 ../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8 #EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180 ../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 Filename: playlist.m3u8. The six different bitrates correspond to six different video streams of the same live event video but at varying quality: • 6434112 (referred to herein as "**6434112 Bandwidth**") • 864048 (referred to herein as "**864048 Bandwidth**") • 403824 (referred to herein as "**403824 Bandwidth**") • 367728 (referred to herein as "**367728 Bandwidth**") • 312832 (referred to herein as "**312832 Bandwidth**") • 249664 (referred to herein as "**249664 Bandwidth**") These different bitrate versions include at least a "low quality stream, a medium quality stream, and a high quality stream." For example, the **6434112 Bandwidth** version can be considered a high-quality stream, the **403824 Bandwidth** version can be considered a medium-quality stream, and the **249664 Bandwidth** version can be considered a low-quality stream. As shown herein, each of the high-quality stream (e.g., the **6434112 Bandwidth** stream), the medium-quality stream (e.g., the **403824 Bandwidth** stream), and the low-quality stream (e.g., the

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		249664 Bandwidth stream) comprise "a group of streamlets encoded at the same respective one of the different bitrates." Each variant playlist includes at least two streamlets (e.g., "at least first and second streamlets"): a **media_1274.ts**" segment and a "**media_1275.ts**" segment. A comparison of the **6434112 Bandwidth**, **403823 Bandwidth**, and **249664 Bandwidth** versions from above shows that each playlist includes segments with these file names. On information and belief, playlists for the other three variants also include these segments. As discussed above, each streamlet corresponds to a portion of the video. Notably, for example, each bitrate version of the **media_1275.ts** segment has a duration of 2.0 seconds (as indicated in each line beginning with "#EXTINF"). Upon information and belief, the Mirror Devices operate in the same or substantially the same way as the Mirror Application. For example, during a test of the Mirror Devices, a first version, a second version, and a third version of the live video were captured. The first version corresponds to a high-

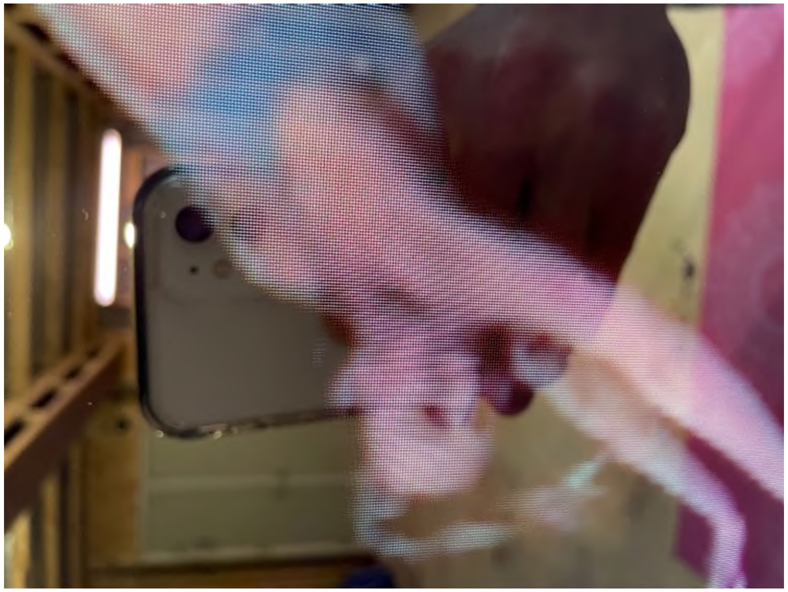
6434112 Bandwidth	403824 Bandwidth	249664 Bandwidth



Claim	Claim Limitation	Example Infringement Evidence
		quality stream, the second version corresponds to a medium-quality stream, and the third version corresponds to a low-quality stream. First version: 

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Second version: 

Claim	Claim Limitation	Example Infringement Evidence
		Third version: 
	wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bit rate of no less than 600 kbps; and	As shown above, "at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bit rate of no less than 600 kbps." At least the high-quality stream (**6326576 Bandwidth**) and one of the medium quality streams (**864048 Bandwidth**) is encoded at a bitrate of not less than 600 kbps as indicated by its "BANDWIDTH" attribute, which signals the upper bound of the overall bitrate for the streamlets in bits per second and is listed at over 6 megabits and 800 kilobits per second.
	wherein the first streamlets of each of the low quality stream, the medium quality stream and the high quality	As shown above, the "first streamlets of each of the low quality stream, the medium quality stream, and the high quality stream each has an equal playback duration and each of the first streamlets encodes the same portion of the live event video at a different one of the different bitrates." As discussed above, each of the **6434112 Bandwidth**, the **403824 Bandwidth**, and **249664 Bandwidth** variant playlists includes a "first streamlet" (e.g., **media_1275.ts** segment). Each of the

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td>stream each has an equal playback duration and each of the first streamlets encodes the same portion of the video at a different one of the different bitrates;</td><td>variant “media_1275.ts” segments has an “equal playback duration” of 2.0 seconds (as indicated in each line beginning with “#EXTINF”) and “encodes the same portion of the live event the video” identified by the “media_1275.ts” segment in different bitrates. Upon information and belief, this is also true for the Mirror Devices as explained above.</td></tr>
<tr><td></td><td>select a specific one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams;</td><td>The non-transitory machine readable instructions of the Mirror Application and the Mirror Devices cause the processor to “select a specific one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams.”

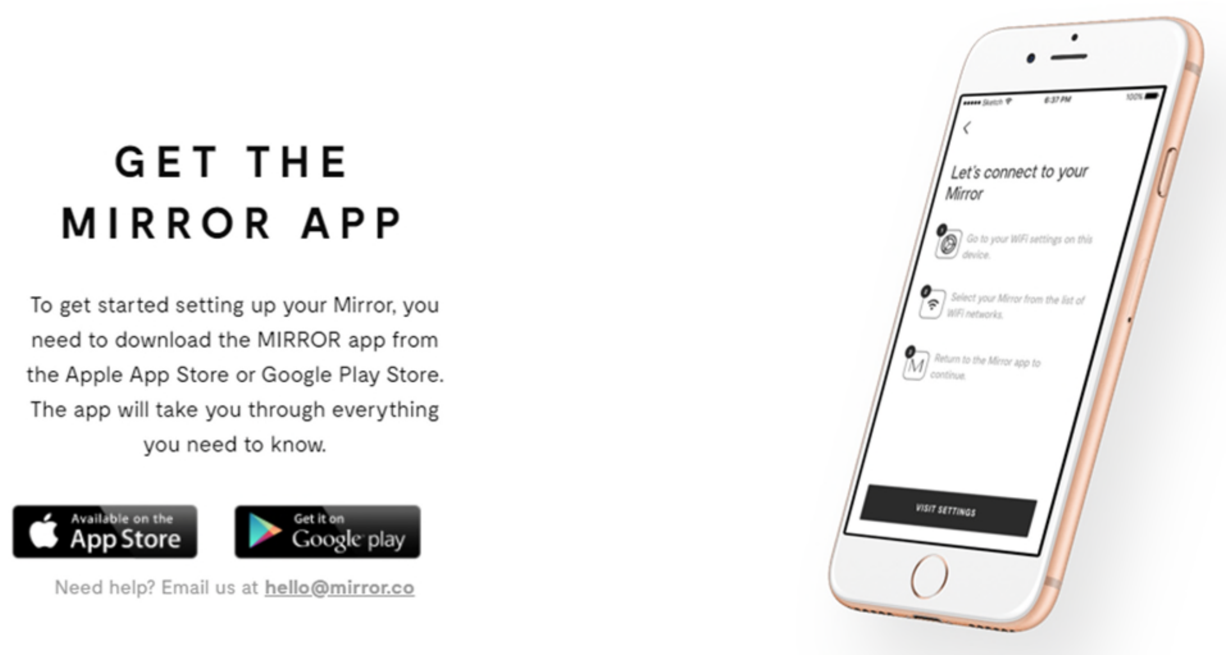
Based upon, at least in part, a determination of the available bandwidth, the Mirror Application and Mirror Devices may determine to “select a higher or lower bitrate version of the stream” and thereby “select a specific one of” the low-quality stream (e.g., the 249664 Bandwidth stream), the medium-quality stream (e.g., the 403824 Bandwidth stream), and the high-quality stream (e.g., the 6434112 Bandwidth stream).

As part of the testing, the Mirror Application was connected to the Internet through the Charles Proxy application. For the instant test, the Mirror Application selects the 403824 Bandwidth stream as indicated by its request for a 403824 Bandwidth playlist (see GET request for d1f65f45_1_1728/chunklist.m3u8) and subsequent request for the 403824 Bandwidth version of the “media_1277.ts” file. When the available bandwidth was reduced during the test, the Mirror Application subsequently selected a different, lower bandwidth version of the stream. Below is an excerpt of the Charles Proxy application “Sequence” listing showing the same.

<table>
<tr><th>Method</th><th>Host</th><th>Path</th></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8</td></tr>
<tr><td colspan="3">..</td></tr>
<tr><td>GET</td><td>wowzaprod102-i.akamaihd.net</td><td>/hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts</td></tr>
</table></td></tr>
</table>

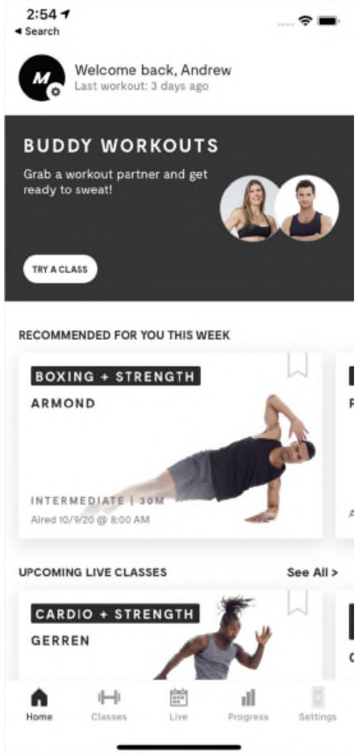
Claim	Claim Limitation	Example Infringement Evidence
		… GET | wowzaprod102-i.akamaihd.net | /hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 … GET | wowzaprod102-i.akamaihd.net | /hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts Upon information and belief, for at least the reasons stated above, the Mirror Devices and the Mirror Application operate in the same or substantially the same way.
	place at least one virtual timeline request for at least one virtual times based on the selected one of the he low quality stream, the medium quality stream, and the high quality stream; and	The non-transitory machine-readable instructions of the Mirror Application cause the processor to "place at least one virtual timeline request for at least virtual time[line] based on the selected one of the … low quality stream, the medium quality stream, and the high quality stream." As shown above, when the Mirror Application has selected the **403824 Bandwidth** (medium quality) version of the stream, the Mirror Application requests the virtual timeline for that selected bandwidth version of the stream. In the Charles Proxy sequence listing below, the Mirror Application requests the virtual timeline (variant playlist) for the **403824 Bandwidth** version of the video: **Method** | **Host** | **Path** GET | wowzaprod102-i.akamaihd.net | /hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 Upon information and belief, and for the reasons set forth herein, the Mirror Devices operate in the same or substantially the same way.
	receive the at least one virtual timeline.	The non-transitory machine-readable instructions of the Mirror Application cause the processor to "receive the at least one virtual timeline." In response to the request for the **403824 Bandwidth** virtual timeline shown above, the virtual timeline is retrieved from the specified file path and sent to the requesting end user station running the Mirror Application. The contents of the response including the **403824 Bandwidth** virtual timeline received by the Mirror Application is shown below.

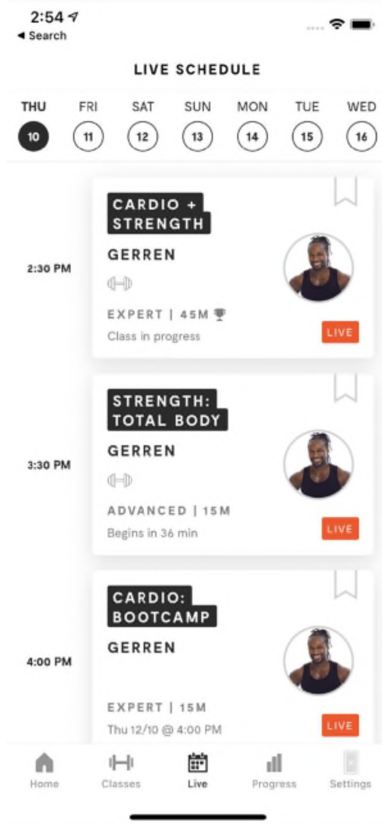
Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1238 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z #EXTINF:2.0, fbd862nq/00000000/media_1238.ts #EXTINF:2.0, fbd862nq/00000000/media_1239.ts #EXTINF:2.0, fbd862nq/00000000/media_1240.ts #EXTINF:2.0, fbd862nq/00000000/media_1241.ts #EXTINF:2.0, fbd862nq/00000000/media_1242.ts #EXTINF:2.0, fbd862nq/00000000/media_1243.ts #EXTINF:2.0, fbd862nq/00000000/media_1244.ts #EXTINF:2.0, fbd862nq/00000000/media_1245.ts #EXTINF:2.0, fbd862nq/00000000/media_1246.ts #EXTINF:2.0, fbd862nq/00000000/media_1247.ts #EXTINF:2.0, fbd862nq/00000000/media_1248.ts Filename: **chunklist.m3u8**
28	A process executable by a content player device to stream a video over a network	The Mirror Application is software that causes "a content player device to stream a video over a network from a server for playback of the video." The Mirror Application is "executable by" end user stations that have a "content player device" and the Mirror Application streams of a selected video program for playback of the video. The streams are obtained over a network.

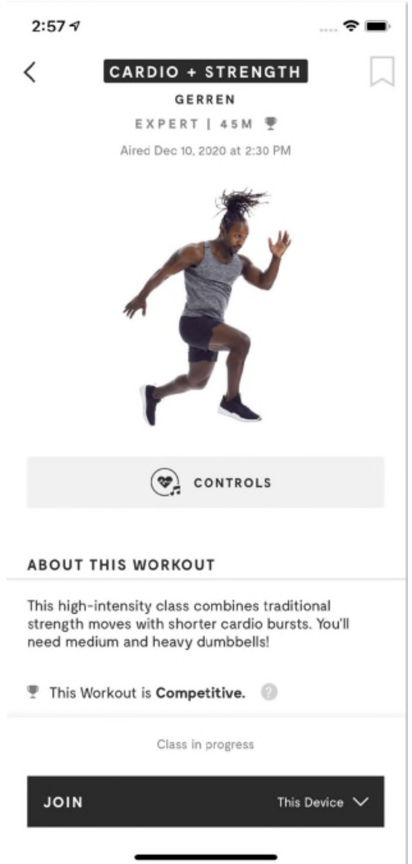
	from a server for playback of the video by the content player device, the process comprising:	The exemplary images in this chart of the Mirror Application are from the Mirror Application running on an Apple iPhone XS (Mirror's iOS Application). In addition, the Mirror Application is available to run on other devices. Unless otherwise noted, each of these devices is an end user station having a "content player device." **MIRROR DIGITAL OVERVIEW** Access all of MIRROR's Live and On Demand Classes 24/7 from your phone, tablet, and smart TV devices. This feature is included for free with the MIRROR Membership and access is available immediately (whether you already have your Mirror or are awaiting delivery). Please note, it is not currently possible to purchase MIRROR Digital if you do not own a Mirror. https://mirror.kustomer.help/en_us/mirror-digital-overview-B1pnVd8UU  https://www.mirror.co/app.

Claim	Claim Limitation	Example Infringement Evidence
		MIRROR APP The MIRROR App allows you to access and customize the Mirror experience. The MIRROR App is available for both iOS and Android! • To access MIRROR content via iOS you'll need a device running iOS 10 or later. • To access MIRROR content via Android, you'll need a device running Android 7 (Nougat) or later. https://mirror.kustomer.help/en_us/mirror-app--S1dDC_tYm.

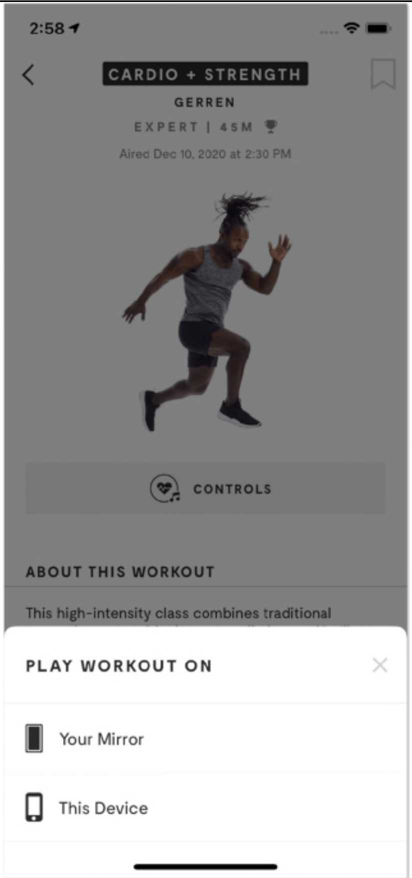
Claim	Claim Limitation	Example Infringement Evidence
		 Source: Apple App Store When launched, the Mirror Application displays a main menu:

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application The main menu of the Mirror Application displays classes that are each a "video." The "Upcoming Live Classes" section of the Mirror Application main menu displays a preview of ongoing and upcoming live video programming.

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting an ongoing live class from the list causes the Mirror Application to display more details regarding the class and provides the user with the option to join the class.

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting the "Join" button for an ongoing live event causes the Mirror Application to provide options to stream the class to a variety of end user stations over the Internet, including the iOS device that the user is using or the separate Mirror device.

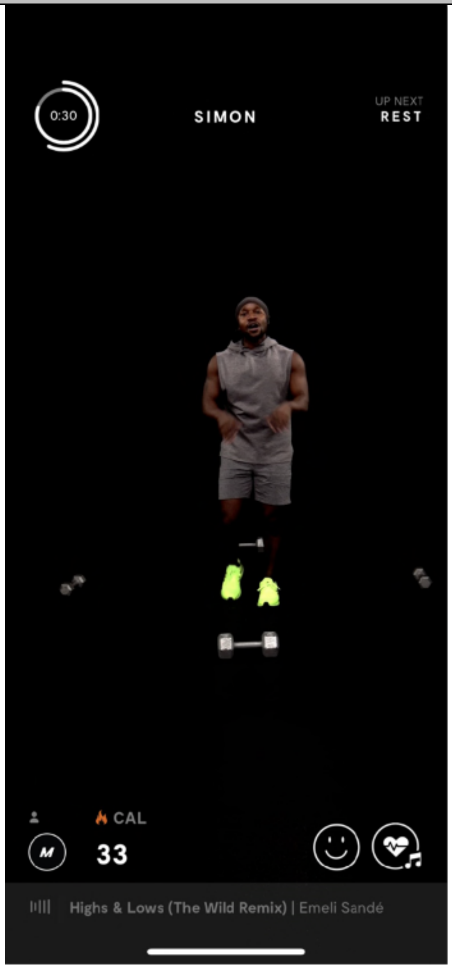
USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application Selecting “Your Mirror” causes the selected live event video and other materials to be streamed on the user’s Mirror device, which is connected to the Internet.

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Alternatively, selecting "This Device" causes the selected workout video and other materials to be streamed on the user's iOS device:

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		 Source: Mirror iOS Application As shown above, Mirror Devices include content player devices to stream a video over a network from a server for playback of the video. The Mirror Devices obtain streams of a selected video program for playback. The streams are obtained over a network.

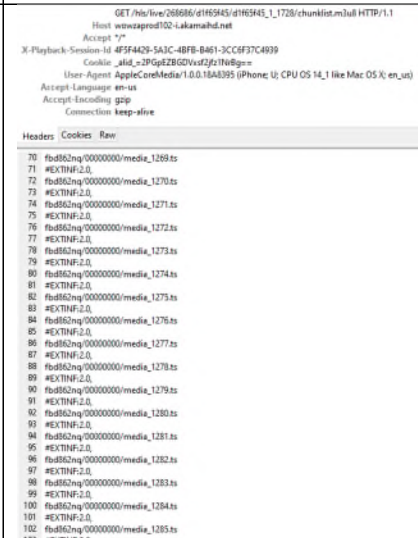
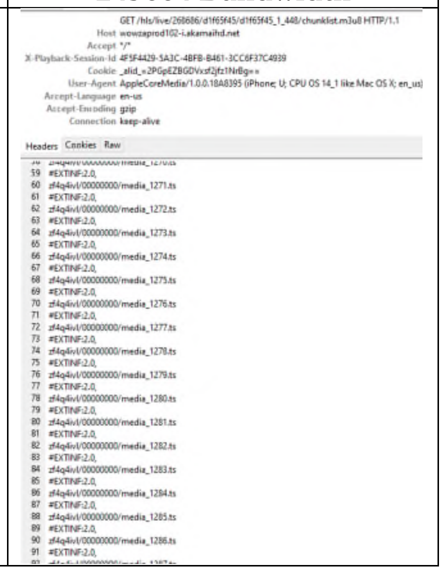
Claim	Claim Limitation	Example Infringement Evidence
	establishing one or more network connections between the content player device and the server, wherein the server accesses a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video;	The Mirror Application and Mirror Devices perform the step of "establishing one or more network connections between the content player device and the server" that "accesses a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates," where each group comprises "at least first and second streamlets" and "each of the streamlets corresponding to a portion of the video." The Mirror Application requires an internet connection. **PRELOAD CLASSES ON MIRROR DIGITAL** You currently cannot preload classes on the MIRROR Digital, however this feature is coming soon! If you are not able to connect WiFi or are in a tough WiFi environment, you can always use cellular data to stream classes on the MIRROR App. Please consult your cell phone provider for questions about data usage and your plan. https://mirror.kustomer.help/en_us/preload-classes-on-mirror-app-H12XPdUUL. To stream a video, such as that shown above, the Mirror Application requests a stream of a selected live event video program via a network connection. When the Mirror Application accesses a selected video program, it requests and receives a playlist file that shows the available versions of the video at different bandwidths and resolutions. For the following test, a live event video was selected. In the test, an iPhone 11 running the Mirror Application makes an HTTPS GET request to "**https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/playlist.m3u8**" for a master playlist named "**playlist.m3u8**" that specifies the available streams and provides links to the playlists for those streams. The following master playlist named "**playlist.m3u8**" is returned.

Claim	Claim Limitation	Example Infringement Evidence

		`#EXTM3U`
		`#EXT-X-VERSION:3`
		`#EXT-X-STREAM-INF:BANDWIDTH=6434112,CODECS="avc1.100.40,mp4a.40.2",RESOLUTION=1920x1080`
		`../../268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=864048,CODECS="avc1.100.41,mp4a.40.2",RESOLUTION=1280x720`
		`../../268686/d1f65f45/d1f65f45_1_2728/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=403824,CODECS="avc1.77.40,mp4a.40.2",RESOLUTION=854x480`
		`../../268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=367728,CODECS="avc1.77.32,mp4a.40.2",RESOLUTION=640x360`
		`../../268686/d1f65f45/d1f65f45_1_1152/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=312832,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=512x288`
		`../../268686/d1f65f45/d1f65f45_1_640/chunklist.m3u8`
		`#EXT-X-STREAM-INF:BANDWIDTH=249664,CODECS="avc1.66.30,mp4a.40.2",RESOLUTION=320x180`
		`../../268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8`
		
		Filename: playlist.m3u8.
		
		This is a master playlist file according to the HLS specification.[2] The playlist shows six versions of the stream, denoted by each #EXT-X-STREAM-INF tag at the following bandwidth:
		• 6434112 (referred to herein as "**6434112 Bandwidth**")
		• 864048 (referred to herein as "**864048 Bandwidth**")
		• 403824 (referred to herein as "**403824 Bandwidth**")
		• 367728 (referred to herein as "**367728 Bandwidth**")
		• 312832 (referred to herein as "**312832 Bandwidth**")
		• 249664 (referred to herein as "**249664 Bandwidth**")
		
		For each of these versions, the master playlist provides a link to a playlist file for the specified version of the selected live event video at a particular bandwidth and resolution, which is called a "variant" in HLS.

[2] RFC 8216 (HLS Live Streaming), Section 4.3.4 (Master Playlist Tags)

USP 10,951,680 to Mirror

<table>
<tr><th>Claim</th><th>Claim Limitation</th><th>Example Infringement Evidence</th></tr>
<tr><td></td><td></td><td>
These different bitrate versions include at least a “low quality stream, a medium quality stream, and a high quality stream.” For example, the 6434112 Bandwidth version can be considered a high-quality stream, the 403824 Bandwidth version can be considered a medium-quality stream, and the 249664 Bandwidth version can be considered a low-quality stream.
As shown herein, each of the high-quality stream (e.g., the 6434112 Bandwidth stream), the medium-quality stream (e.g., the 403824 Bandwidth stream), and the low-quality stream (e.g., the 249664 Bandwidth stream) comprise “a group of streamlets encoded at the same respective one of the different bitrates.” Each variant playlist includes at least two streamlets (e.g., “at least first and second streamlets”): a media_1238.ts” segment and a “media_1239.ts” segment. A comparison of the 6434112 Bandwidth, 403823 Bandwidth, and 249664 Bandwidth versions from above shows that each playlist includes segments with these file names. On information and belief, playlists for the other three variants also include these segments.
<table>
<tr><th>6434112 Bandwidth</th><th>403824 Bandwidth</th><th>249664 Bandwidth</th></tr>
<tr><td></td><td></td><td></td></tr>
</table>
</td></tr>
</table>

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		As discussed above, each streamlet corresponds to a portion of the video. Notably, for example, each bitrate version of the **media_1275.ts** segment has a duration of 2.0 seconds (as indicated in each line beginning with "#EXTINF"). The Mirror Application initially selects the **6434112 Bandwidth** (1080p – high bandwidth) version of the stream and makes a request for the corresponding variant playlist file named "**chunklist.m3u8**." That file with the following contents (a portion of which is shown below) is returned.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:1232
#EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:08.356Z
#EXTINF:2.0,
r4vhrugx/00000000/media_1232.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1233.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1234.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1235.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1236.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1237.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1238.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1239.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1240.ts
#EXTINF:2.0,
r4vhrugx/00000000/media_1241.ts
#EXTINF:2.0,
```

Claim	Claim Limitation	Example Infringement Evidence
		Path: https://wowzaprod102-i.akamaihd.net/hls/live/268686/d1f65f45/d1f65f45_1_4128/chunklist.m3u8 As noted above, the variant playlist file is an HLS playlist. The variant playlist file identifies a plurality of segments or "streamlets" that are part of the **6434112 Bandwidth** group of streamlets. Each line in the file "**chunklist.m3u8**" that begins with "#EXTINF" specifies the length of the segments in seconds (2.0). The line below the #EXTINF entry is the relative location of the video file for the segment (e.g., r4vhrugx/0000000/media_1232.ts). The Mirror Application uses HTTPS GET requests to retrieve the segments of the encoded live event video specified in the file above, and the requested segments are accessed and returned in response to the requests from the Mirror Application. The Mirror Application makes a request for a segment r4vhrugx/00000000/media_1232.ts, and the requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected video. As long as the viewer stays on the selected video and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **6434112 Bandwidth** version). While the example above relates to the **6434112 Bandwidth** group of streamlets, other groups of streamlets are available. If the available bandwidth for the network connection decreases, for example, the Mirror Application will continue to request segments to continue streaming the video, but at one of the lower bandwidths. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variable playlist file for the **403824 Bandwidth** named "chunklist.m3u8." The file is returned with the following contents showing a portion of the **403824 Bandwidth** group of segments for the video being streamed.

Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1238 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z #EXTINF:2.0, fbd862nq/00000000/media_1238.ts #EXTINF:2.0, fbd862nq/00000000/media_1239.ts #EXTINF:2.0, fbd862nq/00000000/media_1240.ts #EXTINF:2.0, fbd862nq/00000000/media_1241.ts #EXTINF:2.0, fbd862nq/00000000/media_1242.ts #EXTINF:2.0, fbd862nq/00000000/media_1243.ts #EXTINF:2.0, fbd862nq/00000000/media_1244.ts #EXTINF:2.0, fbd862nq/00000000/media_1245.ts #EXTINF:2.0, fbd862nq/00000000/media_1246.ts #EXTINF:2.0, fbd862nq/00000000/media_1247.ts #EXTINF:2.0, fbd862nq/00000000/media_1248.ts Filename: **chunklist.m3u8** The Mirror Application then makes the request for fbd862nq/00000000/media_1238.ts. The requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected video program at the **403824 Bandwidth**. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror Application will

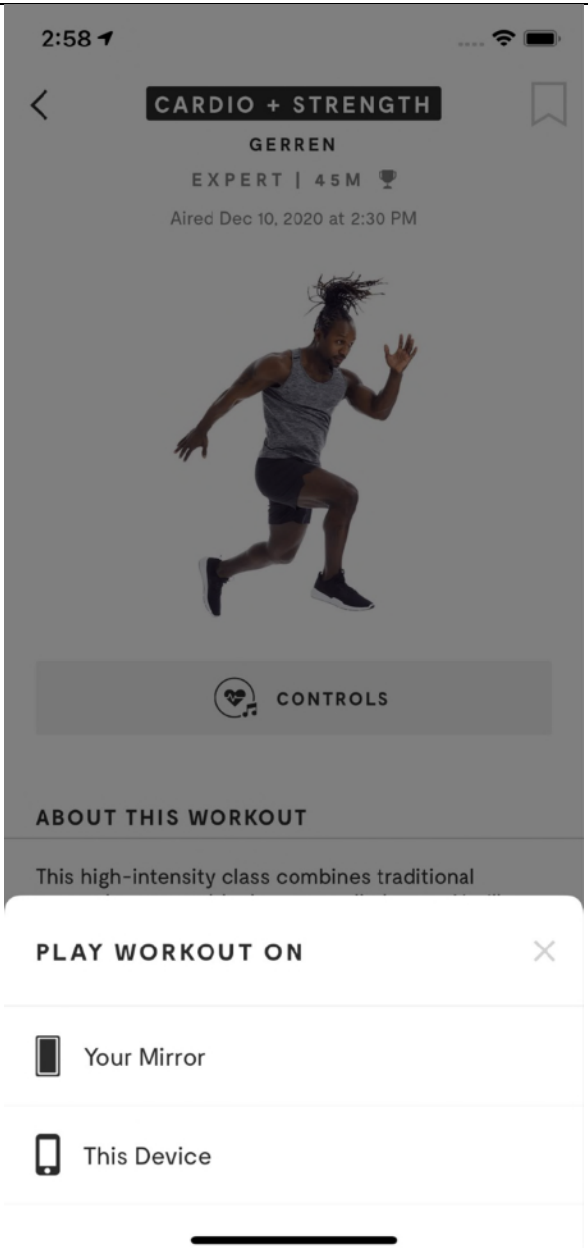
Claim	Claim Limitation	Example Infringement Evidence
		continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **403824 Bandwidth** version). As the bandwidth is further constrained, the Mirror Application makes another request for a lower quality stream. For example, for the current test, the Mirror Application subsequently made a request for the corresponding variable playlist file for the **249664 Bandwidth** named “chunklist.m3u8.” The file is returned with the following contents showing a portion of the **249664 Bandwidth** group of segments for the video being streamed. #EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1241 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:26.354Z #EXTINF:2.0, zf4q4ivl/00000000/media_1241.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1242.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1243.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1244.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1245.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1246.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1247.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1248.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1249.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1250.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1251.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1252.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1253.ts File: **chunklist.m3u8**

Claim	Claim Limitation	Example Infringement Evidence
		The Mirror Application then makes the request for zf4q4ivl/00000000/media_1281.ts. The requested segment is accessed and returned to the Mirror Application, and then the Mirror Application content player plays back the segment to stream the selected video program at the **249664 Bandwidth**. As long as the viewer stays on the stream and the bandwidth is adequate, the Mirror Application will continue to request and receive playlists corresponding to the current, chosen resolution (in this case, the **249664 Bandwidth** version). A portion of a subsequently retrieved playlist's contents are shown below.

Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1245 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:34.354Z #EXTINF:2.0, zf4q4ivl/00000000/media_1245.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1246.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1247.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1248.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1249.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1250.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1251.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1252.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1253.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1254.ts #EXTINF:2.0, zf4q4ivl/00000000/media_1255.ts #EXTINF:2.0, Filename: **chunklist.m3u8** The subsequently retrieved playlist includes additional video segments that were not included in the previous playlist file, for example: "**media_1254.ts**" and "**media_1255.ts.**" The Mirror Application continues to request, receive, and playback successive segments of the video to stream the video. The Mirror Devices also require that users provide an internet connection.

Claim	Claim Limitation	Example Infringement Evidence
		CONNECTION INTERNET — Dual-band 802.11 A/B/G/N Wi-Fi APP — Controlled by iOS or Android companion app HEART RATE — Syncs with Bluetooth™ heart rate monitors, Apple Watches, and Android Wear OS Watches AUDIO — Pairs with Bluetooth™ speakers and headphones https://www.mirror.co/shop/mirror To stream a video, such as that shown above, the Mirror Device requests a stream of a selected video via a network connection. The iOS application provides the interface for interacting with a Mirror Device (i.e., selecting a live or on demand class to stream). After selecting a class, the user selects whether to use the iOS device or Mirror Device to view the content (i.e., stream and participate in the workout).

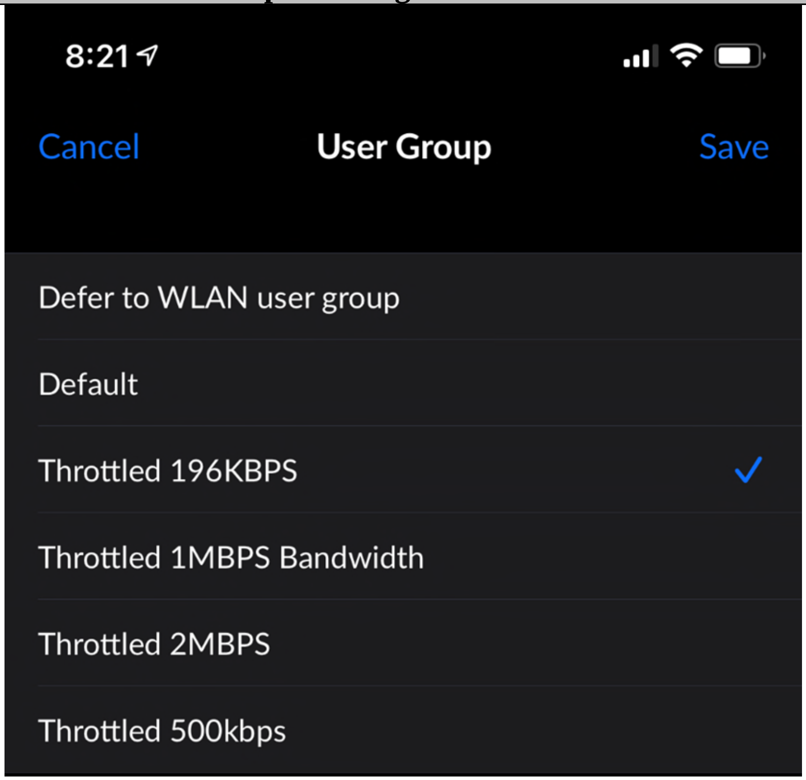
USP 10,951,680 to Mirror



Claim	Claim Limitation	Example Infringement Evidence
		Source: Mirror iOS Application Selecting "Your Mirror" causes the Mirror Device to initiate streaming requests:  For the following test, a live programming event was selected. Based on the test, and upon information and belief, the Mirror Devices operate in substantially the same way as the Mirror Application. For example, when the Mirror Device(s) accesses a selected live event video, the Mirror Device(s) initially selects a first bandwidth version of the stream, makes a request for the segments of the group corresponding to the selected bandwidth version of the live event program, receives segments from the group corresponding to the selected bandwidth version, and then plays the requested and received segments on the Mirror Device content player as shown below.

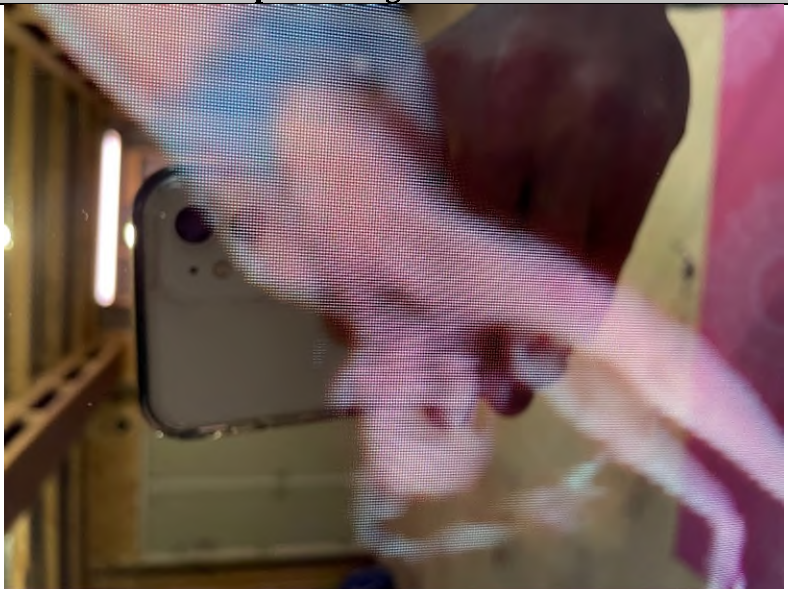
Claim	Claim Limitation	Example Infringement Evidence
		 Other groups of streamlets are also available. For example, for the current test, bandwidth for the Mirror Device was constrained to 196Kbps, which caused the Mirror Device to display a "buffering" message while requesting and receiving a corresponding playlist and streamlets for a second bandwidth version of the live event video as shown below.

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		

Claim	Claim Limitation	Example Infringement Evidence
		 The image resolution for the second bandwidth streamlet requested is noticeably lower quality as indicated by the pixelated edges of the instructor’s body, as shown below.

USP 10,951,680 to Mirror

Claim	Claim Limitation	Example Infringement Evidence
		
	wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and	As shown above, "at least one of the low-quality streams, the medium-quality streams, and the high-quality streams is encoded at a bit rate of no less than 600 kbps." At least the high-quality stream (**6326576 Bandwidth**) and one of the medium quality streams (**864048 Bandwidth**) is encoded at a bitrate of not less than 600 kbps as indicated by its "BANDWIDTH" attribute, which signals the upper bound of the overall bitrate for the streamlets in bits per second and is listed at over 6 megabits and 800 kilobits per second.
	wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same	As shown above, the "first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the live event video in the low quality stream, the medium quality stream, and the high quality stream," and "the first streamlet of the low quality stream ha[s] a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream."

Claim	Claim Limitation	Example Infringement Evidence
	first portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the first streamlet of the low quality stream having a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream;	As discussed above, each of the **6434112 Bandwidth**, the **403824 Bandwidth**, and **249664 Bandwidth** variant playlists includes a "first streamlet" (e.g., **media_1275.ts** segment). Each of the variant "**media_1275.ts**" segments has an "equal playback duration" of 2.0 seconds (as indicated in each line beginning with "#EXTINF") and "encodes the same first portion of the live event the video" identified by the "media_1275.ts" segment in different bitrates. Upon information and belief, this is also true for the Mirror Devices as explained above.
	selecting, by the content player device, a currently selected one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the video;	The Mirror Application and the Mirror Devices perform the step of "selecting, by the content player device, a currently selected one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the live event video." Based upon, at least in part, a determination of the available bandwidth, the Mirror Application and Mirror Devices may determine to "select a higher or lower bitrate version of the stream" and thereby "select a specific one of" the low-quality stream (e.g., the **249664 Bandwidth** stream), the medium-quality stream (e.g., the **403824 Bandwidth** stream), and the high-quality stream (e.g., the **6434112 Bandwidth** stream). As part of the testing, the Mirror Application was connected to the Internet through the Charles Proxy application. For the instant test, the Mirror Application selects the **403824 Bandwidth** stream as indicated by its request for a **403824 Bandwidth** playlist (*see* GET request for d1f65f45_1_1728/chunklist.m3u8) and subsequent request for the **403824 Bandwidth** version of the "media_1277.ts" file. When the available bandwidth was reduced during the test, the Mirror

Claim	Claim Limitation	Example Infringement Evidence
		Application subsequently selected a different, lower bandwidth version of the stream. Below is an excerpt of the Charles Proxy application "Sequence" listing showing the same. **Method** / **Host** / **Path** GET / wowzaprod102-i.akamaihd.net / /hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8 .. GET / wowzaprod102-i.akamaihd.net / /hls/live/268686/d1f65f45/d1f65f45_1_1728/fbd862nq/00000000/media_1277.ts … GET / wowzaprod102-i.akamaihd.net / /hls/live/268686/d1f65f45/d1f65f45_1_448/chunklist.m3u8 … GET / wowzaprod102-i.akamaihd.net / /hls/live/268686/d1f65f45/d1f65f45_1_448/zf4q4ivl/00000000/media_1274.ts Upon information and belief, for at least the reasons stated above, the Mirror Devices and the Mirror Application operate in the same or substantially the same way.
	placing a virtual time request over one or more internet connections from the one or more end user stations to retrieve at least one virtual timeline corresponding to the currently selected one of the low quality stream, the medium quality stream, and	The Mirror Application and Mirror Devices perform the step of "placing a virtual time request over one or more internet connections from the one or more end user stations to retrieve at least one virtual timeline corresponding to the currently selected one of the low quality stream, the medium quality stream, and the high quality stream." As shown above, when the Mirror Application has selected the **403824 Bandwidth** (medium quality) version of the stream, the Mirror Application requests the virtual timeline for that selected bandwidth version of the stream. In the Charles Proxy sequence listing below, the Mirror Application requests the virtual timeline (variant playlist) for the **403824 Bandwidth** version of the video: **Method** / **Host** / **Path** GET / wowzaprod102-i.akamaihd.net / /hls/live/268686/d1f65f45/d1f65f45_1_1728/chunklist.m3u8

Claim	Claim Limitation	Example Infringement Evidence
	the high quality stream; and	Upon information and belief, and for the reasons set forth herein, the Mirror Devices operate in the same or substantially the same way.
	receiving the requested virtual timeline from the server via the one or more network connections.	The Mirror Application and Mirror Devices perform the step of "receiving the requested virtual timeline from the server via the one or more network connections." In response to the request for the **403824 Bandwidth** virtual timeline shown above, the virtual timeline is retrieved from the specified file path and sent to the requesting end user station running the Mirror Application. The contents of the response including the **403824 Bandwidth** virtual timeline received by the Mirror Application is shown below.

Claim	Claim Limitation	Example Infringement Evidence
		#EXTM3U #EXT-X-VERSION:3 #EXT-X-DISCONTINUITY-SEQUENCE:0 #EXT-X-TARGETDURATION:2 #EXT-X-MEDIA-SEQUENCE:1238 #EXT-X-PROGRAM-DATE-TIME:2020-12-15T22:07:20.358Z #EXTINF:2.0, fbd862nq/00000000/media_1238.ts #EXTINF:2.0, fbd862nq/00000000/media_1239.ts #EXTINF:2.0, fbd862nq/00000000/media_1240.ts #EXTINF:2.0, fbd862nq/00000000/media_1241.ts #EXTINF:2.0, fbd862nq/00000000/media_1242.ts #EXTINF:2.0, fbd862nq/00000000/media_1243.ts #EXTINF:2.0, fbd862nq/00000000/media_1244.ts #EXTINF:2.0, fbd862nq/00000000/media_1245.ts #EXTINF:2.0, fbd862nq/00000000/media_1246.ts #EXTINF:2.0, fbd862nq/00000000/media_1247.ts #EXTINF:2.0, fbd862nq/00000000/media_1248.ts Filename: **chunklist.m3u8**